Olli Ohtonen

# LÄMPÖTILAN PID-SÄÄTÖ REAALIAIKAJÄRJESTELMÄSSÄ

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Kevät 2007

Kajaanin ammattikorkeakoulu

**OPINNÄYTETYÖ**
**TIIVISTELMÄ**

| | |
|---|---|
| Koulutusala<br>Tekniikan ja liikenteen ala | Koulutusohjelma<br>Tietotekniikan koulutusohjelma |
| Tekijä(t)<br>Ohtonen Olli | |
| Työn nimi<br>Lämpötilan PID-säätö reaaliaikajärjestelmässä | |
| Vaihtoehtoiset ammattiopinnot<br>Konenäkö ja mittaustekniikka | Ohjaaja(t)<br>Romppainen Pentti |
| | Toimeksiantaja<br>Kajaanin ammattikorkeakoulu |
| Aika<br>Kevät 2007 | Sivumäärä ja liitteet<br>42+1 |


Kajaanin ammattikorkeakoululle hankittiin keväällä 2006 uusi mittaustekniikan laitteisto. Tämä laitteisto oli National Instrumentsin CompactRIO. Lopputyön tarkoituksena oli laitteiston käyttöönotto, lämpötilanohjaussovelluksen rakentaminen ja laitteiston testaus Ziegler-Nicholas-menetelmän PID-säädöllä.

Mittausjärjestelmä koostui CompactRIO-laitteistosta, jossa oli reaaliaikakontrolleri ja FPGA-piiri sekä kaksi I/O-moduulia, lähtöön ja tuloon. Ohjattavana prosessina toimi yksinkertainen lämpötilansäätö, joka oli toteutettu tehovastuksen ja siihen liitetyn Pt100-lämpötila-anturin avulla. Lisäksi järjestelmässä oli kaksi sovitinyksikköä, joilla signaalia vahvistettiin CompactRIO:n tulossa ja lähdössä.

Työtä varten tehtiin kolme ohjelmaa, yksi CompactRIO:n FPGA-piirille, yksi CompactRIO:n reaaliaikakontrollerille ja yksi isäntätietokoneelle. Ohjelmien teossa käytettiin National Instrumentsin LabVIEW-ohjelmistoa.

Tuloksena saatiin toimiva säätöjärjestelmä, jossa tieto lämpöprosessista kulki anturilta kahden laitteen ja ohjelman kautta isäntätietokoneen ohjelmalle. Prosessinohjaustieto siirtyi isäntätietokoneelta takaisin lämpötilansäätöprosessiin.

Testauksessa tehtiin neljä testimittausta, joissa lämpötilan asetusarvo nostettiin arvosta 20 °C arvoon 100 °C, ja seurattiin järjestelmän vastetta eri PID-arvoilla. Testimittausten perusteella Ziegler-Nicholas-menetelmän PI-säätö osoittautui selvästi parhaaksi säätömenetelmäksi tässä järjestelmässä.

CompactRIO-laite on erittäin toimiva tiedonkeruu- ja ohjauslaite. Sillä voidaan mitata useita erilaisia prosesseja, koska laitteeseen voidaan kiinnittää erilaisia I/O-moduuleja, joilla voidaan mitata eri asioita. Tässä sovelluksessa ei nähty CompactRIO:n todellista suorituskykyä, koska ohjattava prosessi oli suhteellisen hidas.



| | |
|---|---|
| Kieli | Suomi |
| Asiasanat | CompactRIO, FPGA, Reaaliaikajärjestelmät, PID |
| Säilytyspaikka | ☒ Kajaanin ammattikorkeakoulun Kaktus-tietokanta<br>☒ Kajaanin ammattikorkeakoulun kirjasto |

Kajaanin ammattikorkeakoulu
University of Applied Sciences

**THESIS**
**ABSTRACT**

| School<br>School of Engineering | Degree Programme<br>Information Technology |
|---|---|
| Author(s)<br>Ohtonen Olli | |
| Title<br>Temperature PID-Control in a Real-Time System | |
| Optional Professional Studies<br>Machine Vision and Measurement Technology | Instructor(s)<br>Romppainen Pentti |
| | Commissioned by<br>Kajaani University of Applied Sciences |
| Date<br>Spring 2007 | Total Number of Pages and Appendices<br>42+1 |


In the spring 2006 new measurement equipment from National Instruments, called CompactRIO, was acquired to Kajaani University of Applied Sciences. This CompactRIO equipment is used to control a temperature process.

The measurement system consists of the CompactRIO equipment, two connection units and a resistor which is heated. The temperature of the resistor was read with the Pt100 temperature sensor. Three control programs were made for this project, one for the CompactRIO's FPGA-chip, one for the CompactRIO's Real-Time Controller and one for the host computer. National Instrument's LabVIEW-software was used to build the control programs.

Four test measurements were made to test the functionality of the system. In the tests the set point of the temperature was raised from 20 °C to 100 °C. Different PID-values were used in the different measurements. After every measurement the response of the system was notice. The results showed that the Ziegler-Nicholas PI-method was the best controlling method in this study.

The CompactRIO equipment is very useful in various measurement tasks. In this case the speed of the temperature change was too slow to show the real performance of CompactRIO.



| | |
|---|---|
| Language of Thesis | Finnish |
| Keywords | CompactRIO, FPGA, Real-Time Control, PID |
| Deposited at | ☒ Kaktus Database at Kajaani University of Applied Sciences<br>☒ Library of Kajaani University of Applied Sciences |

ALKUSANAT

Tämä lopputyö tehtiin Kajaanin ammattikorkeakoululle.

Tahdon kiittää työn ohjaajaa yliopettaja Pentti Romppaista. Häneltä sain apua työn aikana esiintyneissä ongelmissa.

Haluan kiittää myös Ismo Talusta ja Raimo Hurskaista, jotka rakensivat tarvittavat sovitinyksiköt työtäni varten. Lisäksi haluan kiittää Eero Soinista ja Kaisu Korhosta, jotka auttoivat työn loppuvaiheessa kieliasun tarkastamisessa.

> "Ei mikään oo niin viisas kuin insinööri..."
> (Laulu- ja soitinyhtye Eppu Normaali: Suomi Ilmiö)

Kajaanissa 11.5.2007

Olli Ohtonen

SYMBOLILUETTELO

| | |
|---|---|
| ADC | Analog to Digital Converter |
| CLB | Configurable Logic Block |
| CompactRIO | Compact Reconfigurable Input Output |
| DAC | Digital to Analog Converter |
| DFT | Discreet Fourier Transform |
| ECU | Engine Motor Control |
| FPGA | Field Programmable Gate Array |
| IC | Integrated Circuit |
| I/O | Input/Output |
| LabVIEW | Laboratory Virtual Instrumentation Engineering Workbench |
| NI | National Instruments |
| PCB | Printed Circuit Board |
| PID | Proportional Integral Derivate |
| PWM | Pulse Width Modulation |
| VDC | Volts Direct Current |

SISÄLLYS

# 1 JOHDANTO

Keväällä 2006 Kajaanin ammattikorkeakouluun päätettiin yliopettaja Pentti Romppaisen johdolla hankkia uusi mittaustekniikan laitteisto. Laitteisto oli National Instrumentsin (jatkossa NI) CompactRIO-niminen sulautettu järjestelmä. Tästä sain insinöörityön aiheen itselleni. Tavoitteena oli laitteiston käyttöönotto ja lämpötilanohjaussovelluksen tekeminen.

CompactRIO:a käytetään erilaisten järjestelmien ohjaamiseen. Sillä voidaan suorittaa tiedonkeruu prosessista, sen analysointi ja käsittely sekä prosessin ohjaus. CompactRIO koostuu rungosta, jossa on FPGA-piiri, ja reaaliaikakontrollerista. Runkoon kiinnitetään I/O-moduulit, jotka on kytketty FPGA-piiriin tiedon lukua ja kirjoitusta varten. Reaaliaikaprosessorilla voidaan suorittaa tiedon vaativampaa prosessointia.

Insinöörityössä lämpötilanohjausjärjestelmän tiedonkeruu toteutettiin käyttämällä Pt100-lämpötila-anturia. Anturin tulo vahvistettiin käyttämällä tarkoitukseen rakennettua sovitinyksikköä, josta jännitetieto siirrettiin NI9215 I/O -moduulin kautta CompactRIO:lle. Säätöohjelmistot luotiin NI:n LabVIEW-ohjelmalla. Prosessinohjaukseen käytettiin NI9263 I/O -moduulia ja pulssinleveysmodulaatiota (PWM). Ohjauspulssi vahvistettiin välille 0–5 V, ja se ohjattiin tehovastukselle, jota käytettiin lämpöelementtinä sen nopeuden vuoksi.

Jatkossa CompactRIO:a käytetään Kajaanin ammattikorkeakoulussa mittaustekniikan laboraatioissa perehdyttämään oppilaat reaaliaikajärjestelmiin ja FPGA:han.

# 2 TEORIA

## 2.1 Reaaliaikajärjestelmä

Luvussa 2.1 on käytetty NI:n Real-Time Tutorial -materiaalia [1.].

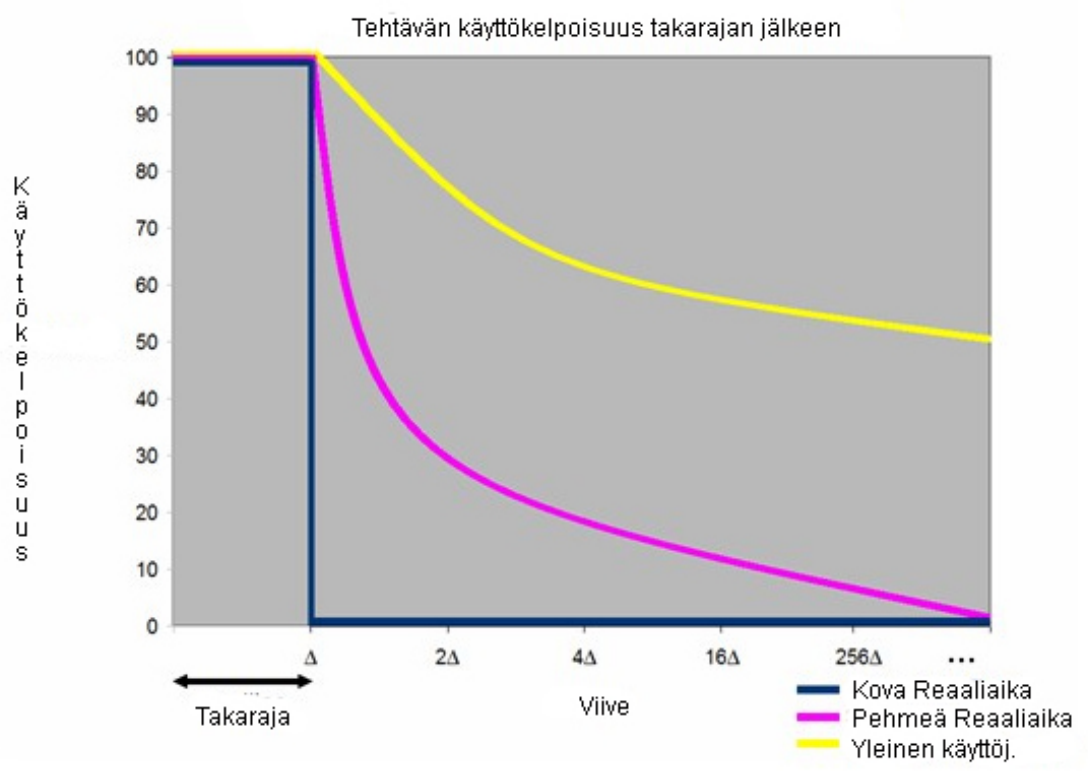
Reaaliaikajärjestelmäksi kutsutaan järjestelmää, jossa tiedon prosessointi tapahtuu deterministisesti. Determinismillä tarkoitetaan sitä, että järjestelmä käyttäytyy ennustettavasti, eli tietty heräte tuottaa tietyn vasteen. Useat testi-, kontrollointi- ja suunnittelusovellukset vaativat reaaliaikaista toteutusta.

### Yleistä reaaliaikajärjestelmistä

Reaaliaikaiset käyttöjärjestelmät saivat alkunsa tarpeesta ratkaista kahta päätyyppiä olevat ongelmat: yksittäiseen tapaukseen tarvittavan vasteen toteuttaminen (event response application) ja suljetun silmukan kontrollisysteemit (closed loop control systems). Yksittäiseen tapaukseen tarvittavan vasteen toteuttaminen vaatii vasteen herätteeseen tietyssä, määrätyssä aikamäärässä. Esimerkkinä tästä on auton turvatyyny. Turvatyynyn tulee laueta tietyssä ajassa kolarin jälkeen, muuten siitä ei ole hyötyä. Suljetun silmukan systeemit taas prosessoivat jatkuvasti herätettä säätääkseen lähtöä. Auton vakionopeudensäädin on hyvä esimerkki tästä. Nopeutta tulee seurata jatkuvasti ja säätää polttoaineen syöttöä sen mukaan. Molemmat edellä esitetyistä tyypeistä vaativat tehtävän toteutusta tietyssä aikamäärässä. Tämän tyypin tapauksia kutsutaan deterministisiksi.

Reaaliaikajärjestelmiä jaotellaan joskus myös ”pehmeään” ja ”kovaan” tyyppiin. Pehmeällä reaaliaikajärjestelmällä tarkoitetaan yleensä sitä, että systeemin käyttökelpoisuus on kääntäenverrannollinen siihen, kuinka kauan sitten tehtävän suorittamisen takaraja ohitettiin. Esimerkkinä tästä on kännykän vastausnäppäimen painaminen: yhteys avautuu pian painamisen jälkeen. Takaraja ei ole kriittinen, ja pienet viiveet hyväksytään. Kovassa reaaliaikajärjestelmässä taas systeemin käyttökelpoisuus on nolla takarajan jälkeen. Auton moottorikontrollin (ECU) täytyy prosessoida sisään tulevat signaalit ja laskea sytytystulppien ajastus takarajaan mennessä. Jos siitä myöhästytään, moottori ei toimi enää kunnolla. Kuvassa 1 on esitetty teh-

tävän käyttökelpoisuus takarajan ylittämisen jälkeen pehmeässä ja kovassa reaaliaikajärjestelmässä sekä yleisessä käyttöjärjestelmässä.




Kuva 1. Tehtävän käyttökelpoisuus eri järjestelmissä takarajan jälkeen

Tietokoneiden käyttöjärjestelmät, kuten Microsoftin Windows tai Macin OS, luovat erinomaisen alustan ei-kriittisten mittausten kehitykselle ja suorittamiselle. Nämä käyttöjärjestelmät on kuitenkin luotu yleiseen käyttöön, ja siksi ne eivät sovi järjestelmiin, joissa vaaditaan determinististä toimintaa tai pidennettyjä käyttöaikoja.

Yleiset käyttöjärjestelmät on optimoitu suorittamaan useita sovelluksia samanaikaisesti. Näin taataan, että kaikki sovellukset saavat hieman prosessointiaikaa. Näiden käyttöjärjestelmien tulee myös vastata etälaitteiden, kuten hiiri ja näppäimistö, keskeytyspyyntöihin. Käyttäjällä ei ole oikeutta päättää siitä, miten asiat ovat prosessorilla hoidettuna. Tästä tuloksena voi olla, että matalamman tärkeysasteen tehtävä ohittaa korkeamman tärkeysasteen tehtävän prosessorilla. Tällaisessa ympäristössä on mahdotonta taata tiettyä vasteaikaa kriittisille sovelluksille.

Reaaliaikakäyttöjärjestelmät antavat käyttäjälle mahdollisuuden priorisoida tehtäviä niin, että kaikista kriittisin tehtävä ottaa prosessorin kontrollin tarvittaessa. Tämä ominaisuus mahdollistaa sovelluksen ohjelmoimisen ennustettavien tuloksien saamiseksi.

Reaaliaikakäyttöjärjestelmiä tarvitaan, kun prosessoria käytetään suljettuun silmukkakontrolliin ja aikakriittisten päätösten tekoon. Nämä sovellukset vaativat ajastettua laskentaa, joka perustuu sisään tulevaan tietoon. Esimerkkinä tästä on I/O-laite, joka näytteistää sisään tulevaa signaalia ja lähettää sen suoraan muistiin. Tämän jälkeen prosessorin täytyy analysoida signaali ja lähettää sopiva vaste I/O-laitteelle. Tässä sovelluksessa ohjelmiston tulee olla mukana silmukassa, ja siksi tarvitaan reaaliaikaista käyttöjärjestelmää. Sillä voidaan varmistaa vaste määrätyssä ajassa. Lisäksi sovelluksissa, joissa tarvitaan pidennettyjä suoritusaikoja käytettään reaaliaikaisia käyttöjärjestelmiä. Sama pätee myös yksinään toimiviin (ns. stand alone) systeemeihin.

Yleisin väärinkäsitys puhuttaessa reaaliaikajärjestelmistä on se, että ne lisäisivät ohjelman suoritusnopeutta. Tämä on totta osassa tapauksista, mutta suurempi hyöty on siitä, että niillä voidaan määritellä tarkasti milloin halutut tehtävät suoritetaan.

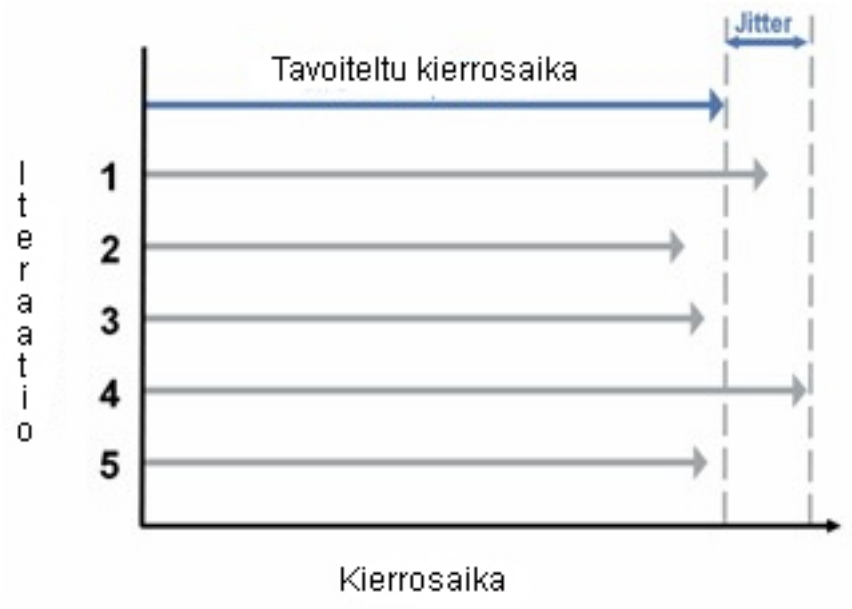
Reaaliaikakontrollointi

Reaaliaikakontrolloinnilla voidaan jatkuvasti seurata ja simuloida fyysistä systeemiä. Reaaliaikakontrollointisovellukset suorittavat jatkuvasti käyttäjän määrittelemää tehtävää tietyssä aikamäärässä. Useimmat reaaliaikakontrollointisysteemit seuraavat fyysistä systeemiä, vertaavat sen viimeisintä tilaa haluttuun tilaan ja sitten ohjaavat systeemiä vertauksen tuloksen perusteella. Tämän silmukan suoritusaikaa kutsutaan silmukan kierrosajaksi. Se vaihtelee systeemin monimutkaisuuden mukaan.

Determinismillä voidaan myös mitata tiettyjen tapausten välisen ajan yhtenäisyyttä. Monet kontrollointialgoritmit, kuten PID, vaativat erittäin determinististä käytöstä. Deterministisyys lisää järjestelmän stabiiliutta.

Kaikissa reaaliaikajärjestelmissä on hieman virhettä, jota kutsutaan englanninkielisellä sanalla jitter. Jitter on toinen tapa mitata reaaliaikajärjestelmän deterministisyyttä. Jitter saadaan lasketuksi minkä tahansa aikaviiveen ja halutun aikaviiveen erotuksena systeemissä. Jitter on se

aika, mikä kuluu vielä sen jälkeen, kun tavoiteltu kierrosaika on loppunut. Kuvassa 2 on esitelty esimerkki jitter-diagrammista.




Kuva 2. Esimerkki jitter-diagrammista

Reaaliaikainen tapausvaste

Reaaliaikaisella tapausvasteella voidaan vastata yksittäiseen tapaukseen tietyssä ajassa. Reaaliaikajärjestelmät takaavat jonkin maksimivastausajan yksittäiseen tapaukseen. Tämä tapaus voi olla joko jatkuva tai satunnainen. Esimerkkinä tästä on turvallisuuden seuranta prosessissa. Jos prosessi menee vaaralliseen tilaan, esimerkiksi lämpötila ylittää raja-arvon, reaaliaikajärjestelmän tulee vastata ”vaaraan” tietyssä, luvatussa ajassa.

Latenssi eli viive on se aika, joka kuluu tapahtumaan vastaamiseen. Se on samanlainen determinismin kanssa reaaliaikajärjestelmissä. Reaaliaikaisella tapausvasteella voidaan taata tietty viive eli latenssi huonoimmassakin tapauksessa.

## 2.2 FPGA

Ohjelmoitavien porttimatriisi- eli FPGA-piirien vahvuus on niiden uudelleenohjelmoitavuus. Piirit on mahdollista mukauttaa jokaiseen sovellukseen sopivaksi. [2.]

Jos sovelluksissa joudutaan useasti turvautumaan bittitason operaatioihin tai siinä käytetään epätavallisia sanapituuksia, FPGA-piireillä saavutetaan huomattavia etuja. Pitkät sanat saadaan käsitellyiksi kerralla, eikä resursseja kulu toisaalta hukkaan lyhyidenkään sanojen osalta. Yksi FPGA-ratkaisun tärkeimmistä eduista verrattuna prosessoreihin on rinnakkaistettavuus, jolla laskentateho voidaan, sovelluksesta riippuen, jopa moninkertaistaa. [2.]

FPGA-piirien sovellukset liittyvät yleensä digitaaliseen signaalinkäsittelyyn, esimerkiksi Fourier-muunnoksiin (DFT), suurten matriisien käsittelyihin ja matemaattisiin algoritmeihin. Tämänkaltaiset operaatiot soveltuvat hyvin ohjelmoitaville logiikkapiireille. [2.]

Luvuissa: Mikä on FPGA, FPGA:n rakenne ja FPGA vs. prosessori on käytetty NI:n materiaalia FPGA-Based Control: Millions of Transistors at Your Command (FAQ) [3.].

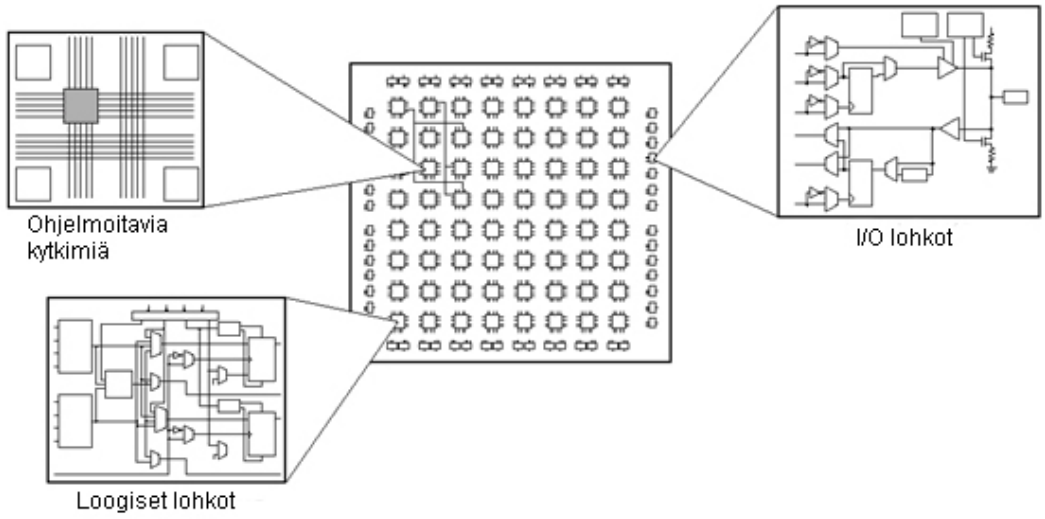
### Mikä on FPGA

FPGA on laite, joka sisältää matriisin uudelleenohjelmoitavia loogisia portteja. Kun FPGA konfiguroidaan, sen sisäiset piirit kytketään siten, että FPGA:n laitteistototeutus vastaa kirjoitettua ohjelmakoodia. Toisin kuin prosessorit, FPGA:t käyttävät laitteistototeutusta logiikan prosessoinnissa, eikä niissä ole käyttöjärjestelmää. Koska FPGA:lla on mahdollista toteuttaa rinnakkaisia operaatioita, prosessien ei tarvitse kilpailla samoista resursseista. Tuloksena tästä on se, että sovelluksen yhden osan suorituskykyyn ei vaikuta toisen prosessin lisääminen, eli FPGA:lla voi olla ajossa monta erillistä kontrollisilmukkaa yhtaikaa erillisillä taajuuksilla.

Toisin kuin kiinteät piirilevyt (PCB:t), joissa on kiinteät laitteistoresurssit, FPGA-piirit voidaan tarvittaessa uudelleenkonfiguroida vielä senkin jälkeen, kun säätöjärjestelmä on rakennettu. FPGA-laitteet tuovat muunneltavuutta, suorituskykyä ja luotettavuutta mittaussovelluksiin.

FPGA:n rakenne

Yksi FPGA voi korvata tuhansia komponentteja käyttämällä miljoonia loogisia portteja yhdellä integroidulla piirillä (IC). FPGA:n sisäiset resurssit käsittävät matriisin konfiguroitavia loogisia lohkoja (CLB), jotka on ympäröity I/O-lohkoilla. Signaalit tuodaan FPGA-matriisiin johdoilla ja ohjelmoitavilla kytkimillä. Kuvassa 3 on esitetty FPGA-piirin rakenne.




Kuva 3. FPGA-piirin rakenne

FPGA vs. prosessori

Kuten prosessoripohjaisia säätöjärjestelmiä, myös FPGA:ita on käytetty kaikenlaisissa teollisuuden säätösysteemeissä, kuten analogisissa prosessinohjauksissa ja logiikassa. FPGA-pohjainen säätöjärjestelmä eroaa kuitenkin merkittävällä tavalla prosessoripohjaisista säätöjärjestelmistä.

Kun säätösovellus käännetään FPGA-laitteelle, tuloksena on optimoitu toteutus, joka mahdollistaa todellisen rinnakkaisen prosessoinnin ja jonka luotettavuus ja suorituskyky on PCB:n luokkaa. Koska FPGA-piirillä ei ole käyttöjärjestelmää, ohjelmaa vastaava piiritoteutus takaa maksimaalisen suorituskyvyn ja luotettavuuden.

Toisaalta FPGA-laitteilla voidaan toteuttaa myös determinististä suljetun silmukan kontrollointia erittäin suurilla kierrostaajuuksilla. Monissa FPGA-toteutuksissa anturit ja I/O-moduulit rajoittavat nopeutta enemmän kuin prosessoitava sovellus FPGA:lla.

Taulukossa 1 on esitelty prosessoripohjaisten ja FPGA-pohjaisten sovellusten eroja.

Taulukko 1. Prosessori vs. FPGA

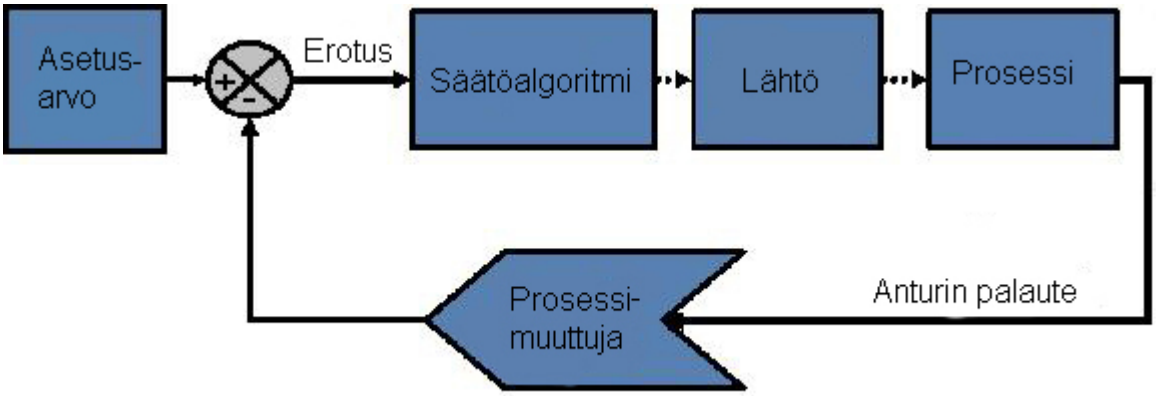
| Prosessoripohjainen | FPGA-pohjainen |
|---|---|
| Suorituskyky rajoittuu 1 kHz:iin. | Suljetun silmukan suorituskyky yli 1 MHz |
| Sarja | Rinnakkainen |
| Hidastuu, kun sovellus kasvaa. | Sovelluksen koolla ei vaikutusta nopeuteen. |
| Käyttöjärjestelmä pyörittää kontrollilogiikkaa. | Kontrollilogiikka tehty itse piirille. |
| I/O moduuleilla yksi käyttötarkoitus. | I/O moduulit uudelleenkonfiguroitavia. |
| Erikoistoteutukset vaativat piirilevyn teon. | Ohjelmalla määriteltävä porttimatriisi. |

## 2.3 Suljetun silmukan järjestelmät

Luvussa 2.3 on käytetty NI:n materiaalia PID Theory Explained [4.]

### Suljettu järjestelmä

Tyypillisessä suljetussa järjestelmässä prosessimuuttuja on parametri, jota säädellään, esimerkiksi lämpötila (°C) tai paine (psi). Anturia käytetään prosessimuuttujan mittaamiseen ja tuottamaan tietoa järjestelmästä. Asetusarvo on arvo, jota kohti prosessimuuttujaa halutaan muuttaa. Asetusarvon ja prosessimuuttujan välistä eroa käytetään suhteena, jonka avulla määritellään ohjaussuure, joka ajetaan järjestelmälle. Esimerkiksi jos prosessimuuttuja on 100 °C ja asetusarvo on 120 °C, voisi kontrollointijärjestelmässä ohjaussuure määritellä käynnistämään lämmittimen. Kun ohjaus määrittelee käynnistämään lämmittimen, järjestelmä lämpenee ja prosessimuuttujan arvo kasvaa. Tällaista kutsutaan suljetuksi järjestelmäksi, koska anturin lukeminen aiheuttaa jatkuvan palautteen ja ohjaussuureen laskennan, jota toistetaan jatkuvasti määrätyllä taajuudella. Tätä on havainnollistettu kuvassa 4.

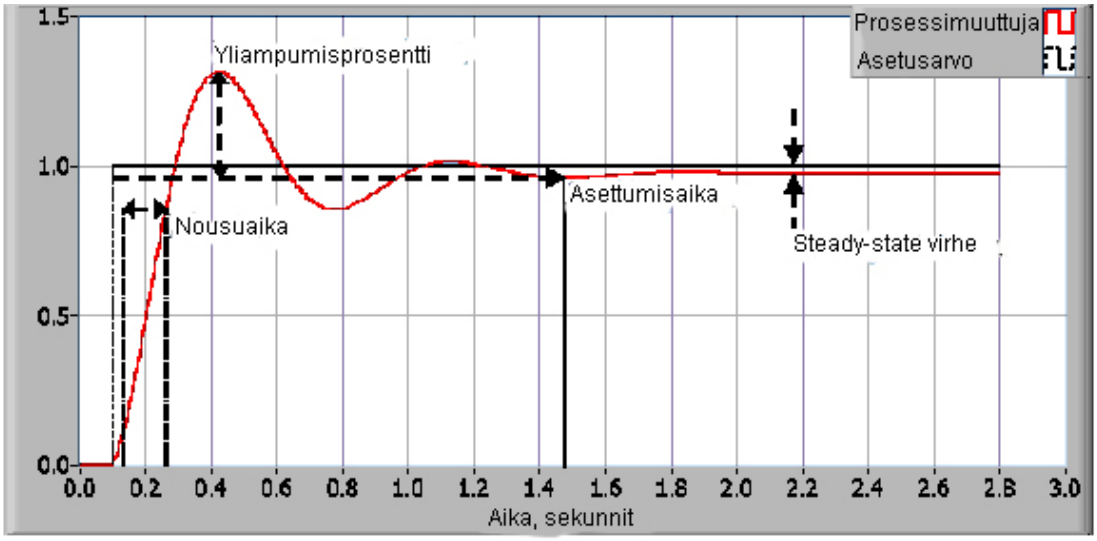



Kuva 4. Tyypillinen suljetun silmukan lohkokaavio

Monissa tapauksissa ohjaussuure ei ole ainut signaali, joka vaikuttaa järjestelmään. Esimerkiksi lämpökammiossa voi olla viileän ilman lähde, joka silloin tällöin puhaltaa kammioon ja vaikuttaa lämpötilaan. Tätä kutsutaan häiriöksi. Yleensä säätöjärjestelmä yritetään rakentaa niin, että häiriön vaikutus prosessimuuttujaan saadaan minimoiduksi.

## Säätöjärjestelmän suunnittelu

Säätöjärjestelmän suunnittelu alkaa yleensä laatuvaatimusten määrittelyllä. Säätöjärjestelmän suorituskyky mitataan yleensä asettamalla askelfunktio asetusarvoon ja mittaamalla prosessimuuttujan vastetta. Nousuaika on aikamäärä mikä systeemiltä kuluu prosessimuuttujan noustessa arvosta 10 % 90 %:iin steady-state- tai loppuarvosta. Yliampumisprosentti on arvo, jonka prosessimuuttuja nousee yli loppuarvon, esitettynä prosenteissa loppuarvosta. Asettumisaika on aika, joka prosessimuuttujalta kuluu, kunnes se asettuu tiettyyn prosenttimäärään (yleensä 5 %) loppuarvosta. Steady-state-virhe on lopullinen ero prosessimuuttujan ja asetusarvon välillä. Näitä asioita on havainnollistettu kuvassa 5. Edellä esitettyjen termien arvot voivat vaihdella teollisuudessa ja kirjallisuudessa lähteen mukaan.

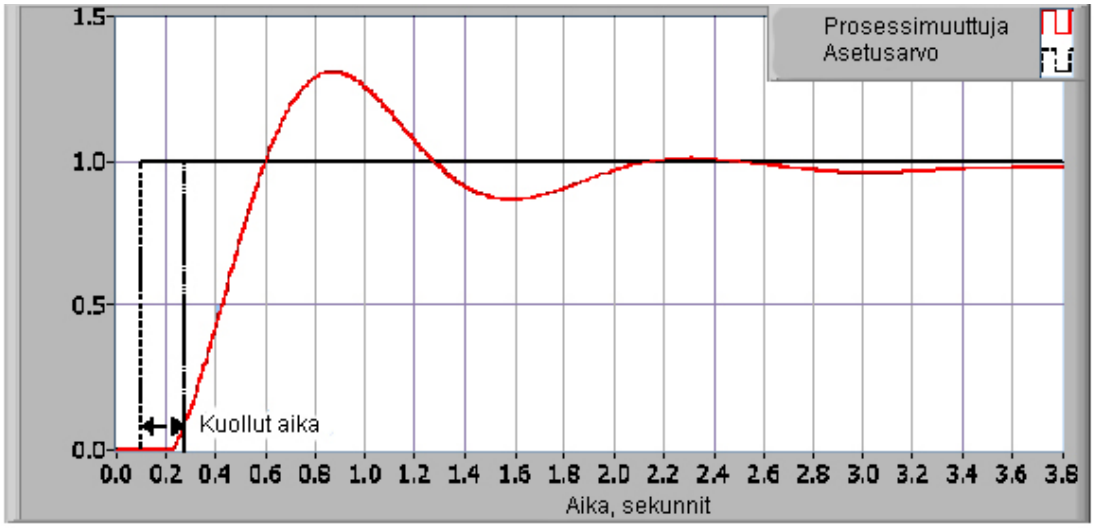



Kuva 5. Tyypillinen suljetun silmukan vaste

Kun on käytetty yhtä tai useampia näistä vaatimuksista säätöjärjestelmän suorituskyvyn määrittelyssä, on hyödyllistä määritellä huonoimman tapauksen olosuhteet, missä säätöjärjestelmän tulisi vielä toimia suunnitteluvaatimusten mukaisesti. Yleensä systeemissä on häiriöitä, jotka vaikuttavat prosessimuuttujaan tai sen laskemiseen. On tärkeää suunnitella säätöjärjestelmä, joka toimii tyydyttävästi vielä huonoimman tapauksen olosuhteissakin. Arvoa, joka kertoo kuinka hyvin säätöjärjestelmä selviytyy häiriöistä, kutsutaan häiriösietoisuudeksi.

Joissakin tapauksissa järjestelmän vaste saattaa muuttua ajan kuluessa tai suhteessa johonkin muuttujaan. Ei-lineaarinen järjestelmä on järjestelmä, joka tuottaa halutun vasteen tietyissä olosuhteissa, mutta ei jossakin toisessa operointipisteessä. Esimerkiksi säiliö, joka on vain osaksi täytetty nesteellä, antaa nopeamman vasteen lämmittimelle kuin säiliö, joka on liki täynnä nestettä. Arvoa kuinka hyvin säätöjärjestelmä sietää häiriöitä ja epälineaarisuuksia, kutsutaan englanninkielisellä sanalla robustness (jyhkeys).

Joissakin systeemeissä on epätoivottua käytöstä, jota kutsutaan nimellä kuollut aika. Kuollut aika on viive prosessimuuttujan muutoksen ja muutoksen havaitsemisen välillä. Esimerkiksi jos nestesäiliössä lämpötila-anturi on sijoitettu kauas kylmän nesteen venttiilistä, anturi ei mittaa muutosta välittömästi, jos venttiiliä avataan ja suljetaan. Kuollut aika voi myös aiheutua systeemistä tai sen lähdöstä, jos se on hidas vastaamaan käskyihin. Esimerkiksi venttiili, joka on hidas avautumaan tai sulkeutumaan. Yleinen kuolleen ajan lähde kemiantehtailla on viive, joka johtuu nesteiden liikkeestä putkien läpi. Kuollut aika säätöjärjestelmän vasteessa on esitetty kuvassa 6.




Kuva 6. Kuollut aika säätöjärjestelmän vasteessa

Kierrosaika on myös tärkeä parametri suljetun silmukan systeemeissä. Aika, joka kuluu kutsusta ohjausalgoritmiin, on kierrosaika. Systeemit, jotka vaihtuvat nopeasti tai omaavat monimutkaisen käytöksen, tarvitsevat suurempia ohjaustaajuuksia.

Suljetun silmukan säätöjärjestelmissä käytetään yleisesti PID-säätöä.

## 2.4 PID-säätö

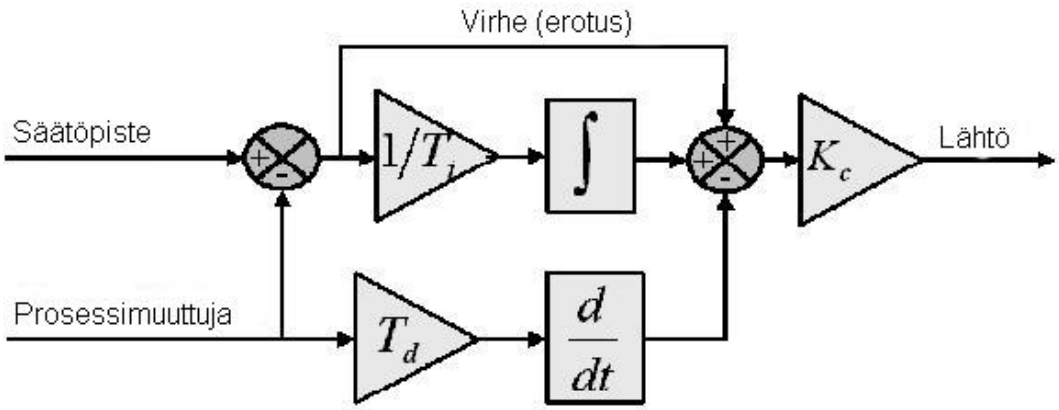
Luvussa 2.4 on käytetty NI:n materiaalia PID Theory Explained [4.]

### Yleistä PID-säädöstä

Tässä esitellään PID-säätöä suljetussa järjestelmässä. PID-säätimen nimi tulee kolmesta termistä, proportional-integral-derivative (suhde-integroiva-derivoiva). PID-säädin on yleisin säätöalgoritmi, jota teollisuudessa käytetään. Se on yleisesti hyväksytty teolliseksi säätimeksi. PID-säätimen suosio johtuu sen vahvasta toimintakyvystä, suuresta käyttöalueesta eri olosuhteissa ja osaksi sen toimivasta yksinkertaisuudesta. Tämä mahdollistaa toimimisen säätimillä yksinkertaisilla ja suoraviivaisilla tavoilla. Kuten nimi kertoo, PID-algoritmi koostuu kolmesta kertoimesta: suhde-, integrointi- ja derivointiosasta, joita muuntelemalla saadaan optimaalinen vaste. PID-säädön perusidea on lukea anturilta tietoa ja laskea lähtö eli ohjaussuure suhde-, integroivan- ja derivoivan kertoimen avulla.

PID-teoria

Seuraavaksi esitellään PID-säädön suhde-, integrointi- ja derivointiparametrit. Kuvassa 7 esitellään tyypillisen PID-säätöjärjestelmän lohkokaavio.




Kuva 7. PID-säätöjärjestelmän lohkokaavio

Suhdevaste

Suhdekomponentti riippuu vain prosessimuuttujan ja asetusarvon erotuksesta. Tätä erotusta kutsutaan erosuureeksi. Suhdevahvistus ($K_c$) määrittää suhteen ohjaussuureen (lähtö) ja erosuureen välillä. Esimerkiksi jos erosuureen arvo on luokkaa 10 ja suhdevahvistuksen arvo luokkaa 5, on tuloksena ohjaussuure, jonka arvo on luokkaa 50. Toisaalta, jos suhdevahvistus on liian suuri, alkaa prosessimuuttuja värähdellä. Jos arvoa $K_c$ vielä tästäkin kasvatetaan, värähtely suurenee ja järjestelmästä tulee epävakaa, ja se voi jopa värähdellä pois kontrollista.

Integrointivaste

Integrointikomponentti summaa erosuureen ajan suhteen. Tuloksena on, että jo pienikin erosuure aiheuttaa integrointikomponentin hitaan kasvun. Integrointivaste kasvaa jatkuvasti ajan suhteen, ellei erosuure ole nolla, eli vaikutuksena on ajaa steady-state-virhe nollaan. Steady-state-virhe on lopullinen erotus prosessimuuttujan ja asetusarvon välillä.

Derivointivaste

Derivointikomponentti aiheuttaa ohjaussuureen (lähtö) pienenemistä, jos prosessimuuttuja nousee voimakkaasti. Derivointivaste on verrannollinen muutoksen nopeuteen prosessimuuttujassa. Derivointiajan ($T_d$) nostaminen aiheuttaa sen, että systeemi reagoi voimakkaammin muutoksiin erosuureessa ja nostaa kontrollointijärjestelmän vasteen nopeutta. Monet käytännölliset järjestelmät käyttävät erittäin lyhyttä derivointiaikaa ($T_d$), koska derivointivaste on erittäin herkkä kohinalle prosessimuuttujassa. Jos prosessimuuttuja on kovin kohinainen tai kontrollisilmukan kierrostaajuus on liian pieni, voi derivointivaste aiheuttaa säätösysteemiin epävakautta.

Eri parametrien arvojen nostaminen vaikuttaa taulukon 2 mukaisesti.

Taulukko 2. Parametrien arvojen nostamisen vaikutus [5.]

| **Parametri** | **Nousuaika** | **Yliampuminen** | **Asettumisaika** | **Steady-State-Virhe** |
|---|---|---|---|---|
| **P** | Pienenee | Suurenee | Pieni muutos | Pienenee |
| **$T_i$** | Pienenee | Suurenee | Suurenee | Eliminoituu |
| **$T_d$** | Pieni muutos | Pienenee | Pienenee | Ei vaikutusta |

PID-järjestelmän säätäminen

Prosessia, jossa asetetaan optimaaliset vahvistukset arvoille P, I ja D ideaalisen vasteen saamiseksi säätösysteemistä, kutsutaan säätämiseksi. Säätämiseen on olemassa useita eri tapoja, joista ”arvaa ja testaa” -menetelmä sekä Ziegler-Nicholas-menetelmä esitellään tässä.

PID-säätimen kertoimet voidaan saada yrityksen ja erehdyksen kautta. Kun käyttäjä ymmärtää jokaisen eri kertoimen merkityksen, tästä menetelmästä tulee helppo. Tässä menetelmässä I- ja D-termit asetetaan aluksi nollaan ja P-termin arvoa nostetaan, kunnes lähtö alkaa värähdellä. P:n arvoa nostettaessa systeemistä tulee nopeampi, mutta systeemin vakaudesta täytyy pitää huolta. Kun P:n arvo on löydetty riittävän nopean vasteen saamiseksi, I-termin arvoa nostetaan värähtelyn pysäyttämiseksi. I-termi pienentää steady-state-virhettä mutta lisää yliampumista. Pieni määrä yliampumista on tarpeen nopeissa systeemeissä, jotta se voisi reagoida muutoksiin välittömästi. I-termiä säädetään niin, että se saavuttaa pienimmän mahdollisen steady-state-virheen. Kun P- ja I-termit on säädetty siten, että saadaan nopea säätösys-

teemi minimaalisella steady-state-virheellä, nostetaan D-termin arvoa kunnes silmukka saavuttaa asetusarvon hyväksyttävän nopeasti. D-termin nosto pienentää yliampumista ja tuottaa suuremman ja vakaamman vahvistuksen, mutta se myös altistaa systeemin herkemmin kohinalle. Jotta saataisiin kuhunkin tilanteeseen sopiva säätö, joudutaan arvoja usein säätämään toisten arvojen kustannuksella.

Ziegler-Nicholas-menetelmä on suosittu PID-säätimen säätömenetelmä. Se on hyvin samantapainen kuin yritys ja erehdys -menetelmä. Menetelmässä parametrit I ja D asetetaan nollaan ja P:n arvoa kasvatetaan, kunnes silmukka alkaa värähdellä. Kun värähtely alkaa, suhdevahvistus $K_c$ ja värähtelyn jakso $P_c$ huomioidaan. Tämän jälkeen P-, I- ja D-kertoimet säädetään taulukon 3 mukaisesti.

Taulukko 3. Ziegler-Nicholas-säätömenetelmä

| **Control** | **P** | **Ti** | **Td** |
|---|---|---|---|
| **P** | 0,5Kc | - | - |
| **PI** | 0,45Kc | Pc/1,2 | - |
| **PID** | 0,60Kc | 0,5Pc | Pc/8 |

# 3 LAITTEET JA OHJELMAT

Työssä käytettiin National Instrumentsin CompactRIO-nimistä tiedonkeruu- ja ohjauslaitteistoa. Ohjelmat luotiin käyttämällä National Instrumentsin LabVIEW 8.0 -ohjelmaa.

## 3.1 Laitteet

### CompactRIO

National Instrumentsin CompactRIO on pieni ja kestävä ohjaus- ja tiedonkeruujärjestelmä, joka on varustettu uudelleenohjelmoitavalla FPGA-teknologialla hyvän suorituskyvyn saavuttamiseksi. CompactRIO sisältää reaaliaikakontrollerin ja uudelleenohjelmoitavan FPGA-piirin. CompactRIO:lla voidaan rakentaa yksinään toimivia (stand-alone) säätö- ja ohjausjärjestelmiä. Kuvassa 8 on esitelty CompactRIO-laite, jossa on reaaliaikakontrolleri ja rungossa 4 I/O-moduulia.

Kuva 8. CompactRIO

## Työssä käytetty CompactRIO-järjestelmä

Työssä käytettiin seuraavanlaista CompactRIO-järjestelmää. Järjestelmä koostuu reaaliaikakontrolleri NI cRIO-9004:stä ja NI cRIO-9104 -rungosta. FPGA-piiri on sijoitettu CompactRIO:n runkoon, jossa on myös paikka 8:lle I/O-moduulille. CompactRIO:ssa on sisäänrakennettu tiedonsiirtomekanismi FPGA-piirin ja reaaliaikaprosessorin välillä. [6.]

## Reaaliaikakontrolleri NI cRIO-9004

Reaaliaikakontrollerissa on 200 MHz:n Pentium-luokan prosessori. Tiedonsiirto ulkoisten laitteiden kanssa on toteutettu 10/100BaseT Ethernet -liitännällä sekä RS232-sarjaportilla. Reaaliaikakontrollerissa on 64 MB:n DRAM-muisti ja 512 MB:n CompactFlash-tallennuskapasiteetti. Tiedonsiirto reaaliaikakontrollerin ja FPGA-piirin välillä on toteutettu paikallisella PCI-väyläliitännällä. [6.]

Reaaliaikakontrollerille voidaan tehdä ohjelma käyttäen LabVIEW Real-Time -moduulia. Reaaliaikakontrollerilla suoritetaan tiedon luku FPGA-piiriltä, tiedon muokkaus, tallentaminen ja halutun tehtävän suorittaminen sekä tiedon siirto takasin FPGA-piirille. [7.]

## Runko NI cRIO-9104

NI cRIO-9104:ssä on FPGA-ydin, jossa on 3 miljoonaa uudellenkonfiguroitavaa porttia. Siinä on 196 KB:n RAM-muisti, ja sen kellotaajuus on 40 MHz, eli yksittäinen kierros kestää 25 ns. [7.]

NI cRIO-9104:ssä on 8 paikkaa I/O-moduuleille. Tieto moduuleista siirtyy FPGA-piirille, jolle voidaan ohjelmoida LabVIEW FPGA -moduulilla ohjelmisto tiedonkeruuta varten. Koska FPGA pystyy käsittelemään vain kokonaislukutyyppistä tietoa, täytyy raskaampi tiedon prosessointi suorittaa reaaliaikakontrollerilla. [8.]

## I/O-moduulit

Jokaisessa CompactRIO:n I/O-moduulissa on sisäänrakennettu signaalinkäsittely, jossa suoritetaan analogia-/digitaalimuunnos (ADC) tai digitaali-/analogiamuunnos (DCA). [9.]

Tässä työssä käytettiin tiedon keräämiseen analogista NI9215 -moduulia. Siinä on neljä kanavaa ja sen käyttöalue on +/- 10 V. NI9215:ssä on 16-bittinen AD-muunnin. [10.]

Lähtönä käytettiin analogista NI9263 -moduulia. NI9263:ssa on neljä kanavaa ja sen käyttöalue on +/- 10 V. NI9263:ssa on 16-bittinen DA-muunnin. [11.]

## 3.2 Ohjelmistot

LabVIEW on National Instrumentsin alusta ja kehitysympäristö graafiselle ohjelmointikielelle. LabVIEW julkaistiin vuonna 1986 Apple Macintoshille ja pian sen jälkeen myös muille käyttöjärjestelmille. Sitä käytetään paljolti elektroniikkatestauksessa ja -simuloinnissa. LabVIEW:illä ja siihen liitetyillä tiedonkeruulaitteilla voidaan kerätä tietoa erilaisista prosesseista. Tietoa voidaan käsitellä lukuisilla eri funktioilla, ja laitteita voidaan ohjata halutusti tämän perusteella. Viimeisin LabVIEW:n versio on 8.2, ja työssä käytettiin versiota 8.0. [12.]

### LabVIEW FPGA -moduuli

LabVIEW FPGA -moduuli laajentaa LabVIEW:n graafista kehitysympäristöä FPGA-laitteille. LabVIEW FPGA -moduulin FPGA-kohteelle erikseen suunnatuilla funktioilla voidaan FPGA-kohteella tehdä helposti ajastusta, liipaisua, synkronointia ja ohjausta.

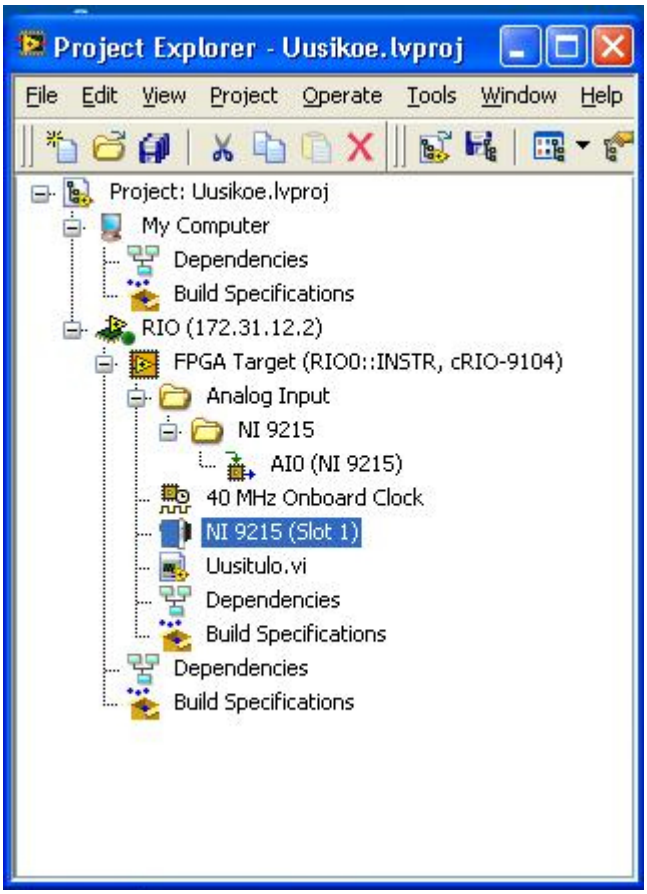
### LabVIEW Real-Time -moduuli

LabVIEW Real-Time -moduuli laajentaa LabVIEW:n käyttömahdollisuudet NI:n laitteistoille, joissa on reaaliaikajärjestelmä. LabVIEW Real-Time -moduulilla voidaan luoda ohjelmakoodeja samassa LabVIEW:n graafisessa ohjelmistoympäristössä ja suunnata ne reaaliaikajärjestelmälle. LabVIEW Real-Time -moduulilla voidaan suorittaa FPGA-kohteelta saadun tiedon raskaampaa prosessointia ja tallentamista.

### Projektit

LabVIEW 8.0 -versiossa tuli uutena ominaisuutena vanhempiin versioihin verrattuna mahdollisuus projektien käyttöön. LabVIEW:n projekti on yhtenäinen kokoelma tiedostoja ja

laitteita, joita käytetään tietyssä sovelluksessa. Projekti esitetään tiedostopuuna, josta näkyy minkä laitteen ”alla” mikäkin ohjelmakoodi on. Projektin käyttö on välttämätöntä CompactRIO:n ja sen tyyppisten laitteiden kohdalla. Laitteissa, joissa tarvitaan useampia ohjelmakoodeja, niiden hallitseminen on helpompaa projektia käyttämällä. Kuvassa 9 on esitelty erään projektin käyttöpaneeli.



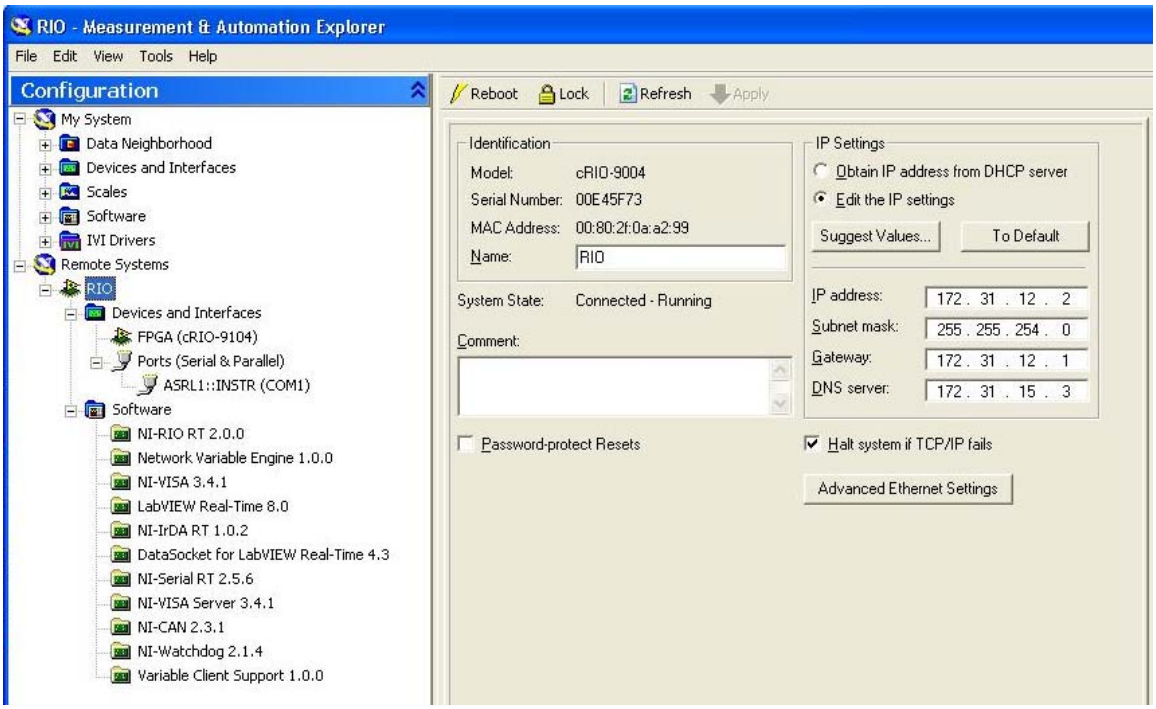

Kuva 9. Erään projektin käyttöpaneeli

## 3.3 Laitteiston ja ohjelmistojen käyttöönotto

### CompactRIO:n konfigurointi

Ensimmäiseksi käytettävälle tietokoneelle asennettiin tarvittavat ohjelmistot, joita olivat LabVIEW 8.0, LabVIEW FPGA -moduuli ja LabVIEW Real-Time -moduuli. Seuraavaksi työssä piti suorittaa CompactRIO:n käyttöönotto. Tämä tapahtui National Instrumentsin Measurement & Automation Explorer (MAX) -ohjelmassa. CompactRIO käynnistettiin, ja

se kytkettiin tietokoneeseen käyttämällä ristiinkytkettyä ethernet -kaapelia. MAX tunnisti CompactRIO:n automaattisesti, jolloin päästiin asettamaan CompactRIO:lle halutut parametrit. Kuvassa 10 on esitetty CompactRIO:n konfigurointi MAX-ohjelmassa.




Kuva 10. CompactRIO:n konfigurointi MAX-ohjelmassa

Projektin luonti

Kun CompactRIO oli konfiguroitu onnistuneesti ja tietokone oli tunnistanut sen, tapahtui projektin luonti. Projekti luotiin wizardia, eli opastusvelhoa, apuna käyttäen. LabVIEW:n aloitusikkunassa valittiin pudotusvalikosta Real-Time Project ja luotiin projekti, jossa oli kaksi silmukkaa sekä isäntäohjelma tietokoneella. Opastusta seuraamalla päästiin lisäämään CompactRIO projektin resursseihin.

CompactRIO:n voi lisätä projektin resursseihin myös manuaalisesti seuraavasti: klikataan projektissa hiiren oikealla näppäimellä projektin nimeä ja valitaan valikosta New → Targets and Devices, ja sieltä valitaan CompactRIO ja sieltä RIO (MAX:issa annettu nimi).

FPGA-kohteen ja I/O-moduulien lisääminen tapahtuu samalla lailla, oli projekti luotu wizardin avulla tai manuaalisesti. Seuraavaksi kerrotaan, kuinka lisääminen tapahtuu.

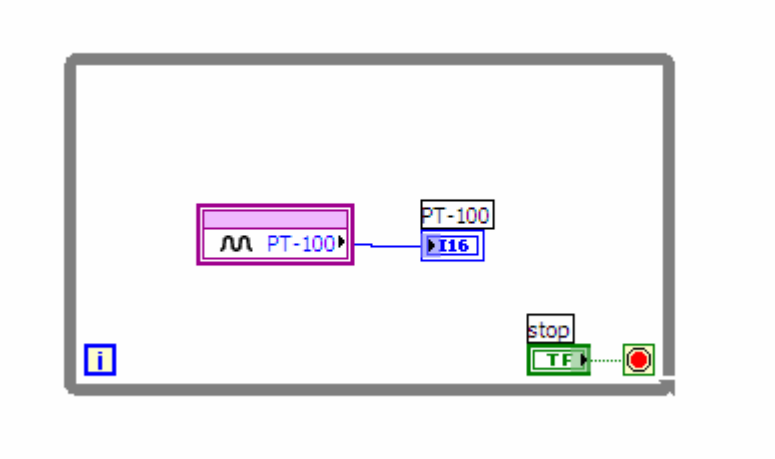
FPGA-kohteen lisääminen projektin resursseihin tapahtuu klikkaamalla projektissa kuvaketta RIO (MAX:issa annettu nimi) hiiren oikealla näppäimellä ja valitsemalla valikosta New→Targets and Devices ja sieltä FPGA Target.

I/O-moduulien lisääminen projektiin tapahtuu seuraavasti: klikataan projektissa hiiren oikealla näppäimellä kuvaketta FPGA Target ja valitaan valikosta New→C Series Modules ja sieltä halutut moduulit.

Lopuksi I/O-moduuliin täytyy lisätä FPGA I/O. Näin saadaan I/O-moduulin kanavat käyttöön. Sen lisääminen moduuliin tapahtuu seuraavasti. Klikataan projektissa I/O-moduulin kuvaketta hiiren oikealla näppäimellä ja valitaan valikosta New→FPGA I/O. Avautuvasta ikkunasta valitaan haluttu moduuli ja siitä haluttu kanava. Tämän jälkeen klikataan Add, jolloin päästään nimeämään kanava halutuksi.

Ohjelmien luonti

Projektin konfiguroinnin jälkeen päästään tekemään itse ohjelmakoodeja. Ohjelmakoodin kohdistaminen FPGA-kohteelle tapahtuu klikkaamalla projektissa hiiren oikealla näppäimellä FPGA Target -kuvaketta ja valitsemalla valikosta New→VI. Tällöin avautuu LabVIEW:in etupaneeli (Front Panel) ja lohkokaavio (Block Diagram), on ovat kohdistettu FPGA-kohteelle. Tällöin käytössä on vain rajoitettu määrä funktioita, koska FPGA ei voi käyttää liukulukuja. Tämän vuoksi esimerkiksi jakolaskuoperaattori ei ole käytössä. Tiedon keruu I/O-moduuleilta tapahtuu ohjelmakoodissa käyttämällä FPGA I/O -funktioita. Kuvassa 11 on esitetty yksinkertaisen tiedonkeruukoodin lohkokaavio. Koodissa käytetään analogista tuloa, joka on nimetty lämpötila-anturin Pt100 mukaan.



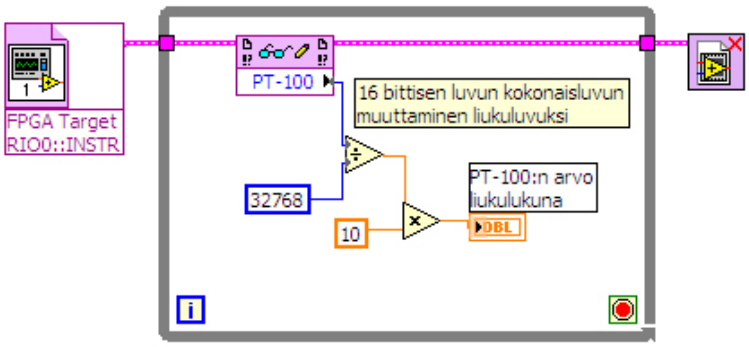

Kuva 11. Yksinkertaisen tiedonkeruukoodin lohkokaavio FPGA-kohteella

Kun ohjelma on kirjoitettu, se käännetään ja ladataan FPGA-kohteelle. Tämä tapahtuu automaattisesti painamalla RUN-näppäintä FPGA-kohteen ohjelman etupaneelissa. Kun ohjelma on ladattu FPGA-kohteelle, se alkaa suorittaa koodin määräämiä tehtäviä.

Reaaliaikakontrollerilla voidaan suorittaa monimutkaisempaa laskentaa ja tiedon prosessointia sekä tallennusta. Wizardi loi reaaliaikakontrollerille automaattisesti ohjelmakoodin, jossa on kaksi silmukkaa. Koodissa on deterministinen ja ei-deterministinen silmukka, joiden välillä tietoa siirretään jaettujen muuttujien (shared variable) avulla.

Ohjelmakoodin manuaalinen luonti reaaliaikakontrollerille tapahtuu seuraavasti. Klikataan hiiren oikealla näppäimellä RIO:ta ja valitaan valikosta New→VI. Tällöin avautuu Lab-

VIEW:n etupaneeli ja lohkokaavio, jotka on kohdistettu RIO:lle. Nyt käytössä ovat kaikki LabVIEW:n funktiot, joten voidaan suorittaa monimutkaisempaa tiedon käsittelyä ja säätöjärjestelmien ohjausta. Tiedon luku FPGA-kohteelta tapahtuu FPGA Interface -funktioilla. Yksinkertainen tiedon luku FPGA-kohteelta sekä tiedon muuttaminen insinööriyksikköihin on esitetty kuvassa 12.




Kuva 12. Tiedon luku FPGA-kohteelta ja sen tyypin muuttaminen

Wizardi loi projektiin ohjelmakoodin myös kuvakkeen My Computer alle. Tässä isäntäohjelmassa suoritetaan prosessitiedon luku reaaliaikakontrollerilta ja ohjaustiedon kirjoitus takaisin reaaliaikakontrolleriin. Tiedon siirto tapahtuu jaetuilla muuttujilla. Tällä ohjelmalla voidaan seurata ja ohjata prosessin kulkua verkon yli.

# 4 SÄÄTÖJÄRJESTELMÄN TOTEUTUS

## 4.1 Suunnittelu

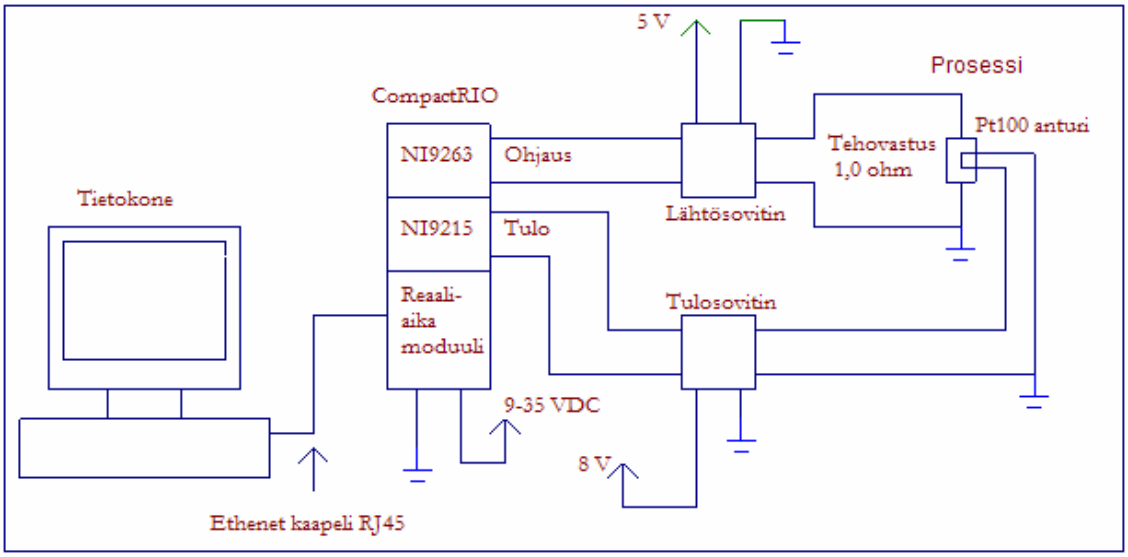
Työssä haluttiin toteuttaa lämpötilan seuranta- ja säätöjärjestelmä. Työtä varten tarvittiin CompactRIO-laite sekä kaksi I/O-moduulia. Ohjattavana lämpöelementtinä käytettiin tehovastusta ja lämpötiloja luettiin lämpötila-anturilla. Pt100-anturia ja signaalin vahvistamista varten tarvittiin kaksi sovitinyksikköä, CompactRIO:n runkoon kiinnitettyjen I/O-moduulien lähtöön ja tuloon.

## 4.2 Rakenne

Säätöjärjestelmä rakennettiin kuvan 13 mukaiseksi. Lämpöelementtinä käytettiin keraamista 1,0 Ω tehovastusta, johon lämpötila-anturi Pt100 oli kiinnitetty metallilankakiinnikkeellä. Pt100:n johdot kiinnitettiin tulosovittimeen, joka ”ajoi” Pt100-anturille tasavirtaa, jotta anturin yli oleva jännite saatiin mitattua. Tulosovitin myös vahvisti anturin yli olevan jännitteen n. 20-kertaiseksi. Tulosovittimen teholähteenä oli verkkomuuntaja, joka antoi sille käyttöjännitteen. Tulosovittimen lähtö kytkettiin CompactRIO:n NI9215 I/O -moduuliin, joka oli kytketty FPGA-piirille.

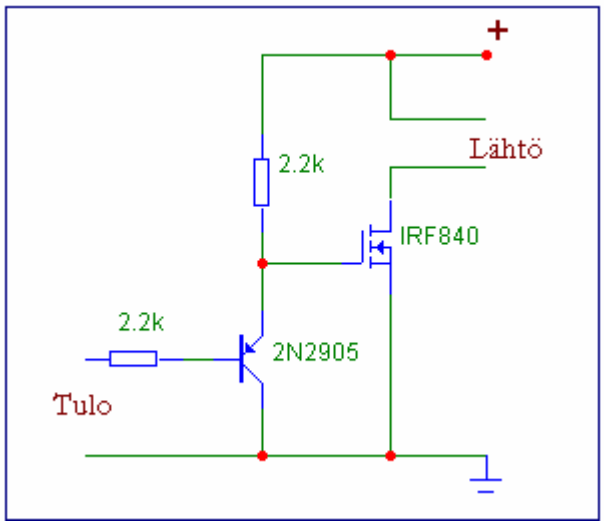
FPGA-piirille oli tehty tiedonkeruuohjelma, jolla signaali luettiin. FPGA-piiriltä signaali siirrettiin reaaliaikakontrollerille, jossa tietotyypin- ja lämpötilanmuunnokset sekä PID-säätö suoritettiin ohjelmallisesti. Reaaliaikakontrollerilta prosessimuuttujan arvo siirrettiin jaettua muuttujaa käyttämällä tietokoneen isäntäohjelmalle. Isäntäohjelmalla säädettiin asetusarvo ja PID-vahvistukset, sekä seurattiin prosessin kulkua ja PID-säädön vaikutuksia siihen graafeilta. Asetusarvo ja PID-vahvistusarvot siirrettiin jaetuilla muuttujilla takaisin reaaliaikakontrollerille. Reaaliaikakontrollerilla tehtiin PID-säätö haluttujen parametrien avulla ja tuloksena saatiin arvo, jota käytettiin ilmaisemaan PWM-signaalin työjakso. Laskettu PWM-signaalin työjakso siirrettiin reaaliaikakontrollerilta FPGA-piirille, jossa pulssinleveysmodulaatio toteutettiin ohjelmallisesti. PWM-signaali siirrettiin FPGA-piiriin kytkettyyn NI9263 I/O -moduuliin.

I/O-moduuli oli ohjelmoitu antamaan 5 voltin signaali, joka johdettiin lähtösovittimelle. Lähtösovitin vahvisti signaalin tehoa samalla (0—5 V) jännitevälillä. Sille otettiin käyttöjännite laboratoriotehonlähteestä. Lämpöelementtinä käytetty tehovastus oli kytketty lähtösovittimen lähtöön. Lähtösovitinta käytettiin, jotta saatiin vastukselle riittävän suuri virta.



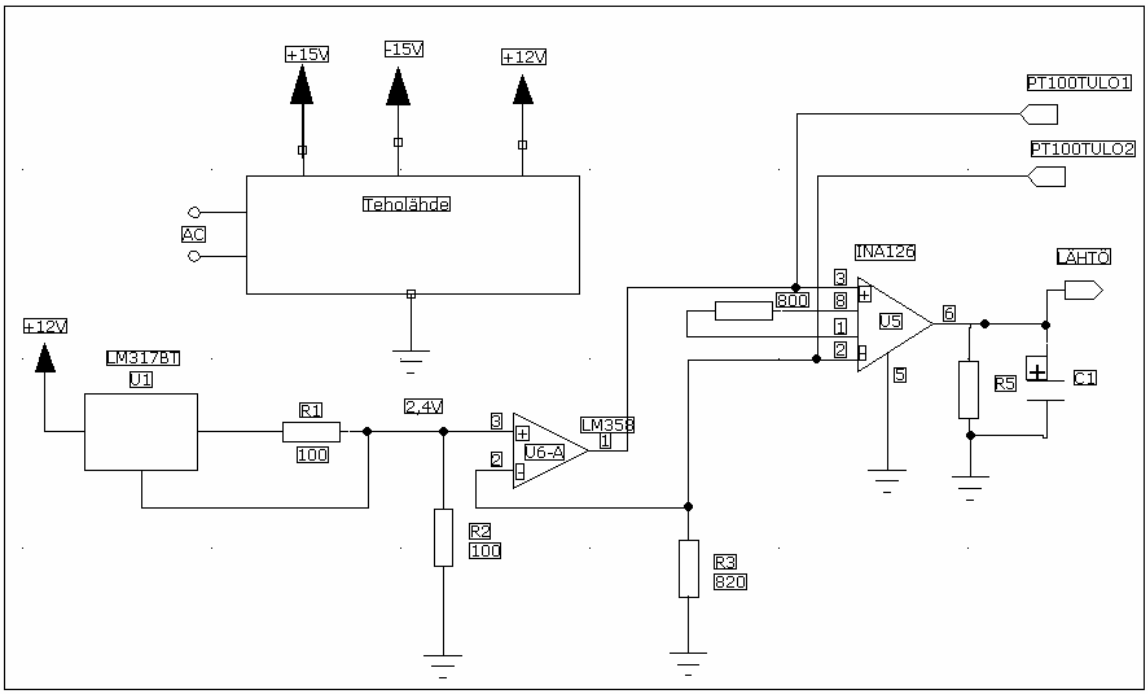

Kuva 13. Periaatekuva mittauslaitteistosta

Lähtö- ja tulosovittimen rakentaminen toteutettiin Kajaanin ammattikorkeakoulun toimesta. Lähtösovittimen piirikaavio on esitetty kuvassa 14, ja tulosovittimen piirikaavio on esitetty kuvassa 15.




Kuva 14. Lähtösovittimen piirikaavio

Koska CompactRIO:n I/O –moduuli ei anna riittävää tehoa vastuksen lämmittämiseen, on moduulin ja vastuksen välissä käytettävä lähtösovitinta. Näin vastukselle saadaan ”ajettua” riittävän suuri virta.




Kuva 15. Tulosovittimen piirikaavio

Tulosovitin ajaa Pt100-lämpötila-anturille vakiovirtaa, jonka avulla saadaan mitattua anturin yli oleva jännite. Tulosovitin myös vahvistaa signaalin n. 20-kertaiseksi.

Pt100

Pt100 on lämpötila-anturi, jonka resistanssi muuttuu lämpötilan vaikutuksesta. Pt100:n resistanssi lämpötilassa 0 °C on 100 Ω, jonka jälkeen resistanssi kasvaa 0,385 Ω jokaista nostettua astetta kohti. Pt100:n resistanssi halutussa lämpötilassa saadaan laskettua kaavalla (1).

$$R_t = R_0 * (1 + \alpha * t), \qquad (1)$$

missä

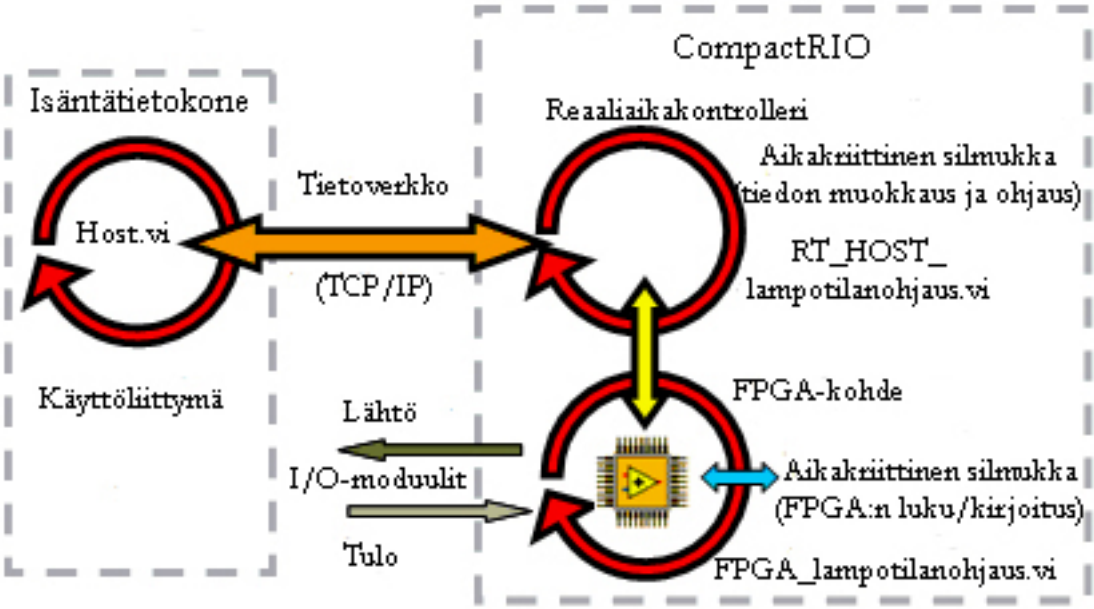
$R_t$ on resistanssi halutussa lämpötilassa

$R_0$ on resistanssi referenssilämpötilassa

α on resistanssin lämpötilakerroin $3{,}9083*10^{-3}$ / K

## 4.3 Ohjelmat

CompactRIO:lle tehtiin 2 kappaletta ohjelmia ja niihin 3 kappaletta aliohjelmia. Lisäksi tietokoneelle luotiin yksi isäntäohjelma. NI cRIO-9104 FPGA -kohteelle luotiin **FPGA_lampotilanohjaus.vi**-ohjelma tiedonkeruuta ja ohjausta varten. Reaaliaikakontrolleri NI cRIO-9004:lle luotiin **RT_HOST_lampotilanohjaus.vi**-ohjelma tiedon muuntamista ja käsittelyä varten. Tässä ohjelmassa oli 3 aliohjelmaa. **16bit_to_Dec.vi**-aliohjelma tehtiin 16-bittisen tiedon muuttamiseksi desimaaliluvuksi (jännitteeksi). **PT-100_to_Temp.vi**-aliohjelma tehtiin jännitteen muuttamiseksi lämpötilaksi. **Pulssisuhteen_muutos.vi**-aliohjelma tehtiin pulssisuhteen kääntämiseksi. Tietokoneelle luotiin **host.vi**-ohjelma prosessin kontrollointia varten. Ohjelmien sijoittumista CompactRIO:lle ja tietokoneelle on esitelty kuvassa 16. Lisäksi liitteessä 1/1 on esitetty vuokaavio koko ohjelmasta.



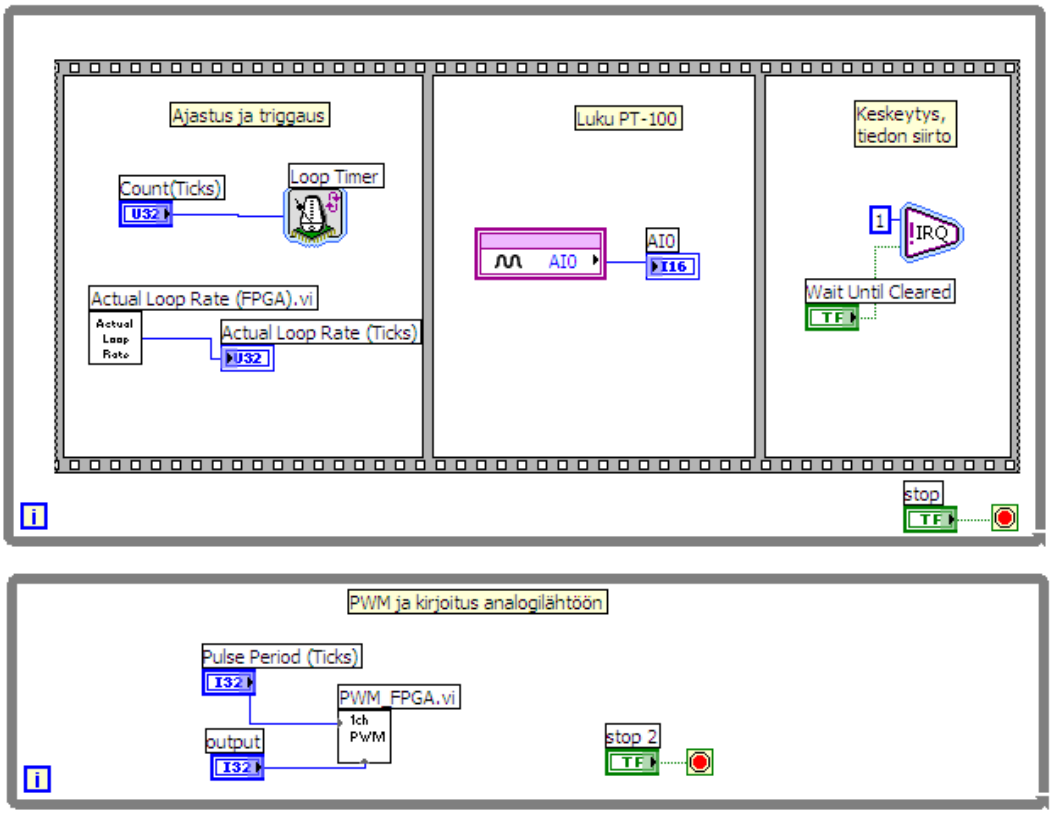

Kuva 16. Ohjelmien sijoittuminen järjestelmässä

### 4.3.1 NI cRIO-9104 FPGA-kohteen ohjelma: FPGA_lampotilanohjaus.vi

NI cRIO-9104 FPGA -kohteen FPGA_lampotilanohjaus.vi-ohjelman etupaneeli on esitetty kuvassa 17a ja lohkokaavio on esitetty kuvassa 17b.


Kuva 17a. Ohjelman FPGA_lampotilanohjaus.vi etupaneeli

Etupaneelissa on indikaattorit ylemmän kierroksen todelliselle kierrosajalle ja Pt100-anturin lukuarvolle. Kontrolleilla päästään säätämään viivettä ylemmässä silmukassa ja voidaan antaa parametreja **FPGA_PWM.vi** aliohjelmalle. Näillä voidaan testata aliohjelman toimivuutta. Lisäksi etupaneelissa on stop-kontrollit molemmille silmukoille.



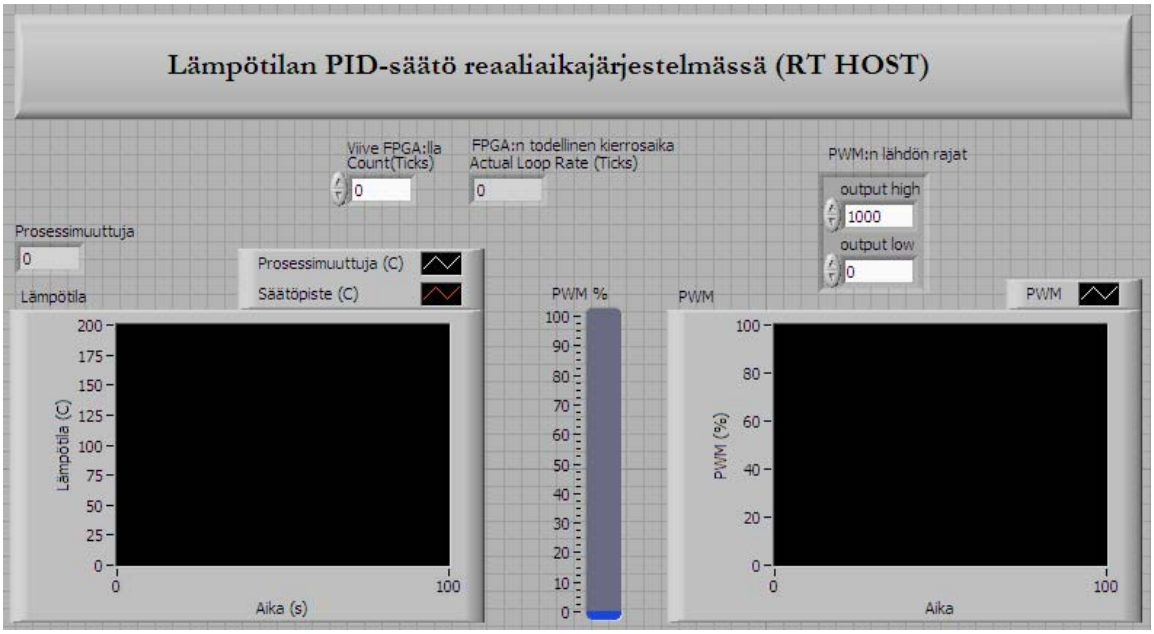

Kuva 17b. FPGA_lampotilanohjaus.vi ohjelman lohkokaavio

Ohjelmassa on kaksi eri taajuudella toimivaa silmukkaa. Ylemmän silmukan sekvenssirakenteen ensimmäisessä kehyksessä tapahtuu ylemmän silmukan ajastus ja silmukan kierrosnopeuden lukeminen aliohjelmalla Actual Loop Rate (FPGA).vi. Keskimmäisessä kehyksessä tapahtuu tiedon luku analogisen I/O-moduulin ensimmäisestä kanavasta (AI0). Viimeisessä kehyksessä tapahtuu tiedon siirto keskeytystä käyttäen reaaliaikakontrollille, mutta tätä ominaisuutta ei käytetty ohjelman testausvaiheessa.

Alemmassa silmukassa annetaan pulssinleveysmodulaation parametrit, pulssin kesto ja pulssin työaika. Parametrit siirretään PWM_FPGA.vi-aliohjelmaan, missä tapahtuu kirjoitus NI9263 I/O -moduulille.

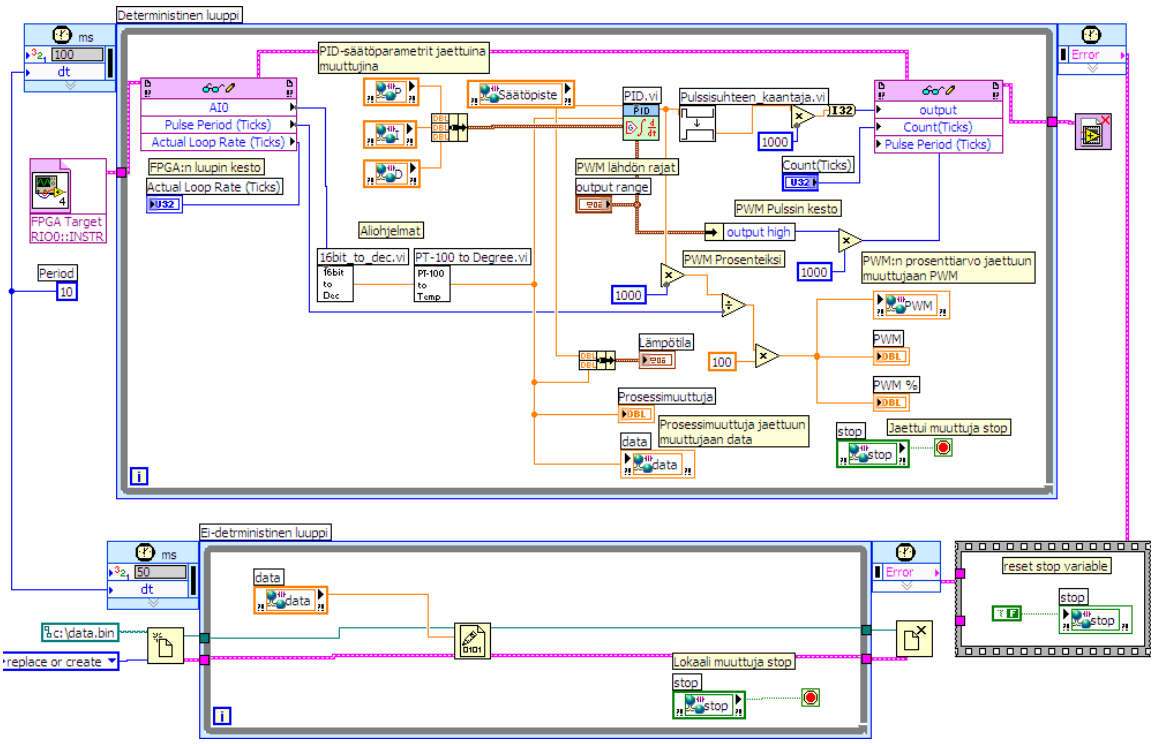
### 4.3.2 NI cRIO-9004 reaaliaikakontrollerin ohjelma: RT_HOST_lampotilanohjaus.vi

Kuvassa 18a on esitetty RT_HOST_lampotilanohjaus.vi-ohjelman etupaneeli ja kuvassa 18b ohjelman lohkokaavio.




Kuva 18a. Ohjelman RT_HOST_lampotilanohjaus.vi etupaneeli.

Etupaneelissa on kaksi graafia, joista oikeanpuoleisessa nähdään prosessimuuttujan ja asetusarvon arvot lämpötilana (ºC). Etupaneelissa on myös kontrolli PID-ohjauksen lähdön ylä- ja alarajaa varten sekä kontrolli viiveen säätämistä varten FPGA-kohteella.




Kuva 18b. RT_HOST_lampotilanohjaus.vi:n ohjelman etupaneeli ja lohkokaavio

Ohjelmassa on kaksi silmukkaa, joista ylempi on aikakriittinen, deterministinen silmukka. Siinä tapahtuu tiedon luku FPGA-kohteelta sekä tiedon muokkaus. Alempi silmukka on ei-deterministinen silmukka, ja siinä tapahtuu tiedon tallentaminen. Tätä ominaisuutta tosin ei käytetty ohjelman testausvaiheessa.

Ylemmässä silmukassa suoritetaan tiedon siirto FPGA-kohteelta ja sen muuntaminen desimaaliluvuksi (jännitteeksi) aliohjelmalla 16bit_to_dec.vi ja jännitetiedon muuttaminen lämpötilaksi aliohjelmalla PT-100_to_degree.vi. Nämä aliohjelmat esitetään myöhemmin.

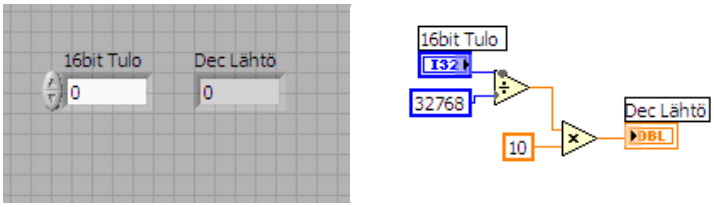
Kirjastofunktio PID.vi:llä suoritetaan PID-säätö prosessimuuttujalle halutuilla PID-vahvistuksilla. PID.vi:n lähtö kerrotaan luvulla 1000, jotta saadaan PWM-pulssin kestoksi riittävän pitkä aika. Yhden silmukan kestoaika FPGA-kohteella on 25 ns. Kun tämä kerrotaan luvulla 1000000 (1000 * 1000), niin saadaan PWM-signaalin kestoajaksi 25 ms eli taajuus on 40 Hz. Lähtö muutetaan tietotyypiksi I32. Tämän jälkeen arvo siirretään FPGA-piirille.

Prosessimuuttujan arvo siirretään jaettuna muuttujana host.vi-ohjelmalle. Asetusarvo, PID-vahvistukset ja stop-napin painallus taas siirretään host.vi-ohjelmalta jaettuina muuttujina RT_HOST_lampotilanohjaus.vi -ohjelmaan.

Seuraavaksi esitetään ohjelmassa RT_HOST_lampotilanohjaus.vi käytetyt aliohjelmat.

Aliohjelma 16bit_to_Dec.vi

Kuvassa 19 on esitetty aliohjelma 16bit_to_Dec.vi, joka muuttaa 16-bittisen tiedon desimaaliluvuksi.




Kuva 19. Aliohjelman 16bit_to_Dec.vi:n etupaneeli ja lohkokaavio

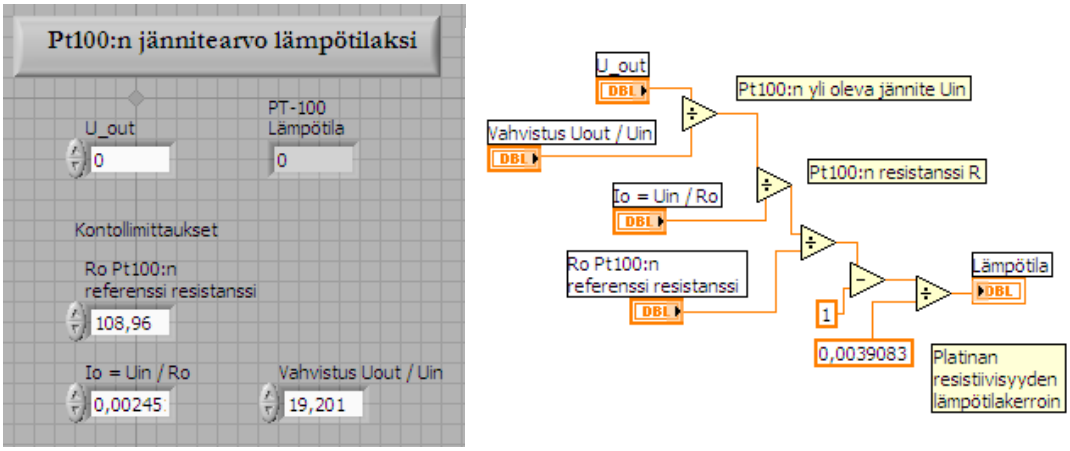
Aliohjelmassa 16-bittinen jännitetieto muutetaan desimaaliluvuksi kaavan (2) mukaan

$$U = \frac{16bit.luku}{32768} * 10{,}0V\,. \qquad (2)$$

Luvut tulevat I/O-moduuli NI9215:n ominaisuuksista. Moduuli on 16-bittinen ja sen referenssijännite on 10 V. [13.]

Aliohjelma PT-100_to_Degree.vi

Kuvassa 20 esitetään aliohjelma PT-100_to_Degree.vi, jossa jännitetieto muutetaan lämpötilaksi.




Kuva 20. Aliohjelman PT-100_to_Temp.vi:n etupaneeli ja lohkokaavio

Lohkokaaviossa esitetty laskenta ja arvot on saatu seuraavaksi esitetyillä mittauksilla ja laskuilla. Pt100-lämpötila-anturin resistanssi mitattiin huoneenlämmössä nelipistemittauksella digitaalisella HP 34401A -mittarilla. Resistanssin referenssiarvoksi saatiin

$R_0$=108,96 Ω.

Tämän jälkeen anturi kytkettiin tulosovittimeen ja tulosovitin kytkettiin muuntajan kautta verkkovirtaan. Samalla digitaalisella HP 34401A -mittarilla mitattiin jännite $U_{in}$ lämpötila-anturin yli. Tulokseksi saatiin

$U_{in}$=267,14 mV.

Mittarilla mitattiin myös jännite $U_{out}$ tulovahvistimen lähdöstä

$U_{out}$=5,1294 V.

Vahvistus A saadaan kaavalla (3),

$$A = \frac{U_{out}}{U_{in}}, \tag{3}$$

josta saadaan

A=19,201.

Kaavalla (4) saadaan laskettua virta $I_0$,

$$I_0 = \frac{U_{in}}{R_0}, \tag{4}$$

josta saadaan

$I_0$=2,4517 mA.

Kuvan 19 lohkokaaviossa suoritetaan laskenta, josta saadaan lämpötila t.

Jännite $U_{in}$ (jännite Pt100-anturin yli) saadaan laskettua kaavalla (5)

$$U_{in} = \frac{U_{out}}{A}. \tag{5}$$

Jännitteen $U_{in}$ ja virran $I_0$ avulla lasketaan Pt100:n resistanssi kaavan (6) mukaisesti

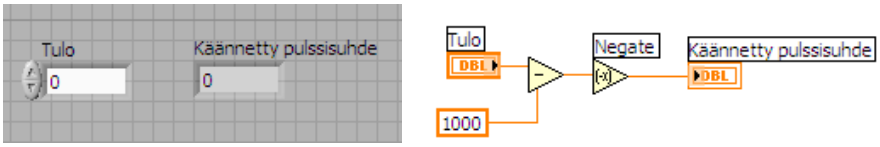
$$\frac{U_{in}}{I_0} = R. \tag{6}$$

Soveltamalla kaavaa (1), saadaan resistanssin R avulla laskettua lämpötila t kaavan (7) mukaisesti

$$t = \frac{R/R_0 - 1}{\alpha}. \tag{7}$$

Aliohjelma Pulssisuhteen_kaantaja.vi

Aliohjelma Pulssisuhteen_kaantaja.vi on esitetty kuvassa 21.

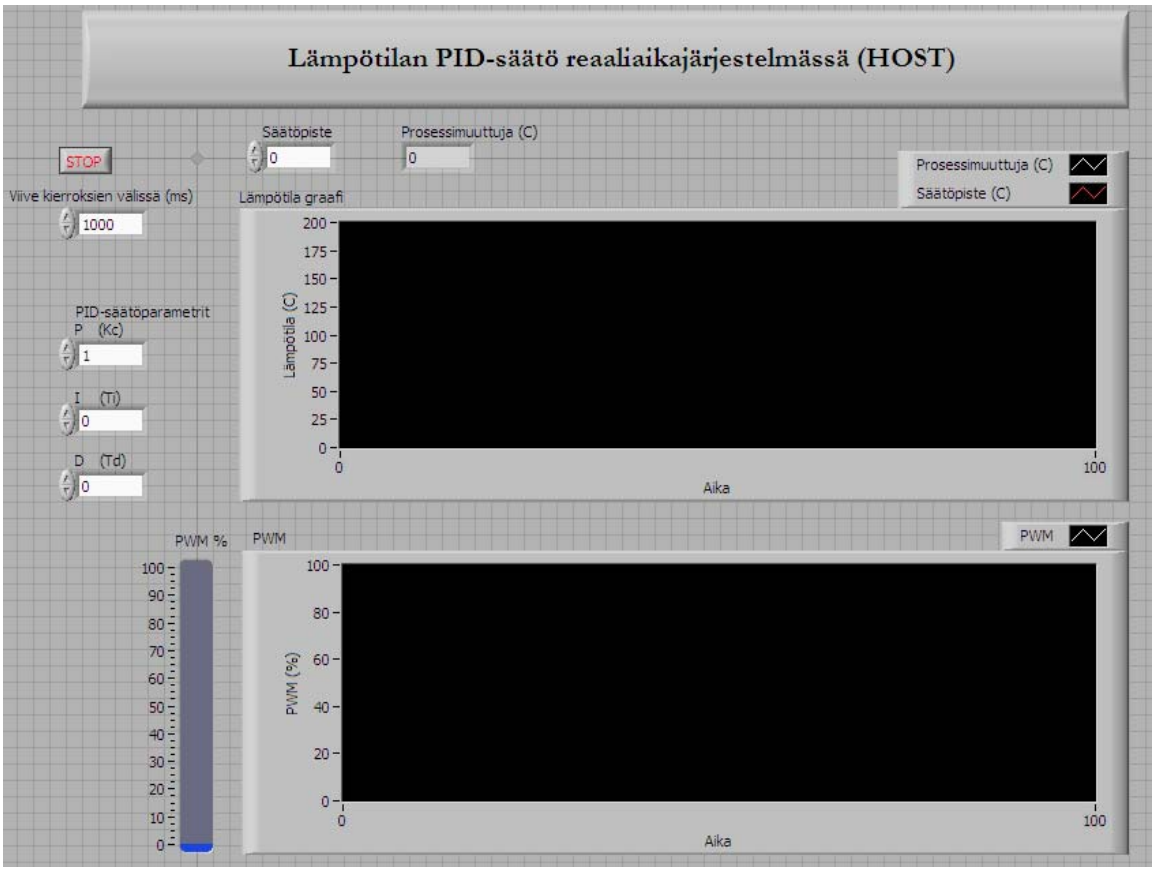



Kuva 21. Aliohjelma Pulssisuhteen_kaantaja.vi:n etupaneeli ja lohkokaavio

Aliohjelmassa suoritetaan pulssisuhteen muutos, koska lähtösovitin toimii seuraavasti. Jos lähtövahvistimen tulossa PWM-pulssin työjakso on 80 %, niin sen lähdössä se on 20 %. Tämän takia pulssisuhde joudutaan kääntämään ohjelmallisesti aliohjelmalla Pulssisuhteen_kaantaja.vi. Aliohjelma toimii seuraavasti: koska tulo on välillä 0—1000, siitä vähennetään luku 1000. Erotus viedään funktion ”Negate” läpi, joka vaihtaa luvun etumerkin. Näin saadaan käännetty pulssisuhde, joka viedään aliohjelman lähtöön.

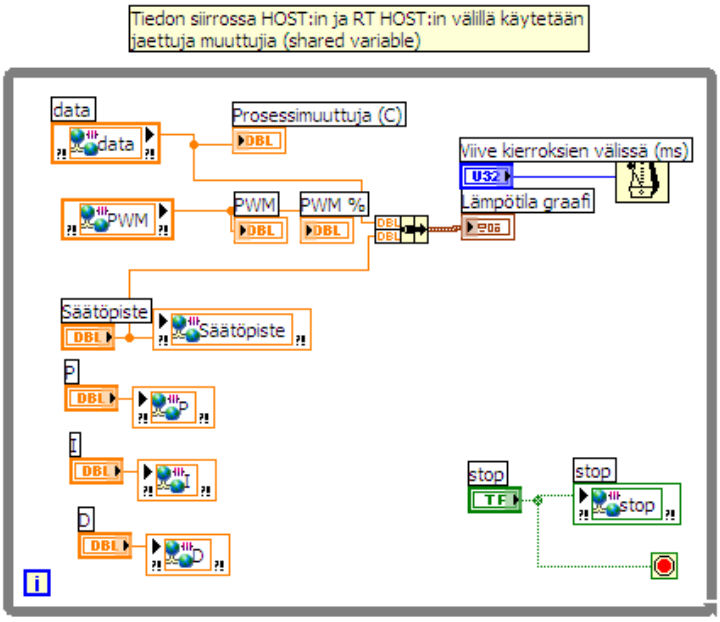
### 4.3.3 Host.vi-isäntäohjelma tietokoneelle

Kuvassa 22a on esitelty ohjelman host.vi etupaneeli ja kuvassa 22b sen lohkokaavio.




Kuva 22a. Ohjelman host.vi etupaneeli

Host.vi-ohjelmalla kontrolloidaan lämpötilaprosessin käyttäytymistä. Ohjelman etupaneelissa on kontrolli nimeltä ”säätöpiste”. Kontrollilla säädetään asetusarvo, johon prosessin lämpötila halutaan. Etupaneelissa on myös kaksi graafia, joista ylempi ”Lämpötila graafi” esittää asetusarvon (säätöpiste) ja prosessimuuttujan arvon muodossa ºC. Alempi ”PWM”-graafi esittää pulssinleveysmodulaation arvon prosentteina. Graafin vieressä on indikaattori ”PWM %”, joka esittää PWM:n hetkellisen arvon prosentteina graafisena palkkina. Ohjelman etupaneelissa on kontrollit PID-säädön P-, I- ja D-parametrien säätämistä varten. Etupaneelissa on myös kontrolli, jolla voidaan säätää silmukan kierrosnopeutta. Oletukseksi on asetettu 1000 ms. ”Stop”-painikkeen tila siirtyy jaettuna muuttujana ohjelmalle RT_HOST_lampotilanohjaus.vi. Painikkeen painaminen lopettaa ohjelmien host.vi sekä RT_HOST_lampotilanohjaus.vi. ajon.




Kuva 22b. Ohjelman host.vi lohkokaavio

Ohjelman lohkokaaviossa suoritetaan prosessimuuttujan luku ohjelmalta RT_HOST_lampotilanohjaus.vi jaetulla muuttujalla ”data”. PID-säätöarvot sekä asetusarvo siirretään ohjelmaan RT_HOST_lampotilanohjaus.vi jaetuilla muuttujilla.

# 5 SÄÄTÖJÄRJESTELMÄN TESTAUS, TULOKSET JA NIIDEN TARKASTELU

## 5.1 Säätöjärjestelmän testaus

Säätöjärjestelmän testaus suoritettiin käyttämällä Ziegler-Nicholas-menetelmää. Testauksessa tehtiin neljä mittausta seuraavan protokollan mukaan. Asetusarvo nostettiin arvosta 20 ºC arvoon 100 ºC, ja prosessimuuttujan arvoa seurattiin 180 sekunnin ajan kymmenen sekunnin välein. Ensimmäisessä mittauksessa tutkittiin, mikä on arvon $K_c$ kriittinen vahvistus. Kriittisen vahvistuksen kohdalla prosessimuuttuja alkaa värähdellä. Kriittinen vahvistus $K_c$ etsittiin käyttämällä vain PID-säädön P-parametria. P-parametrin arvoa nostettiin, kunnes prosessimuuttujan arvo alkoi värähdellä. Kun kriittinen vahvistus löydettiin, säädettiin P-, PI- ja PID-arvot käyttämällä taulukon 2 arvoja.

## 5.2 Tulokset mittauksista

$K_c$:n kriittiseksi vahvistukseksi saatiin 100. Loput mittaukset tehtiin arvoilla, jotka on esitetty taulukossa 4.

Taulukko 4. Mittauksissa käytetyt arvot.

| | **P** | **Ti** | **Td** |
|---|---|---|---|
| **Kriittinen** | 100 | - | - |
| **P** | 50 | - | - |
| **PI** | 45 | 50 s | - |
| **PID** | 60 | 30 s | 7,5 s |

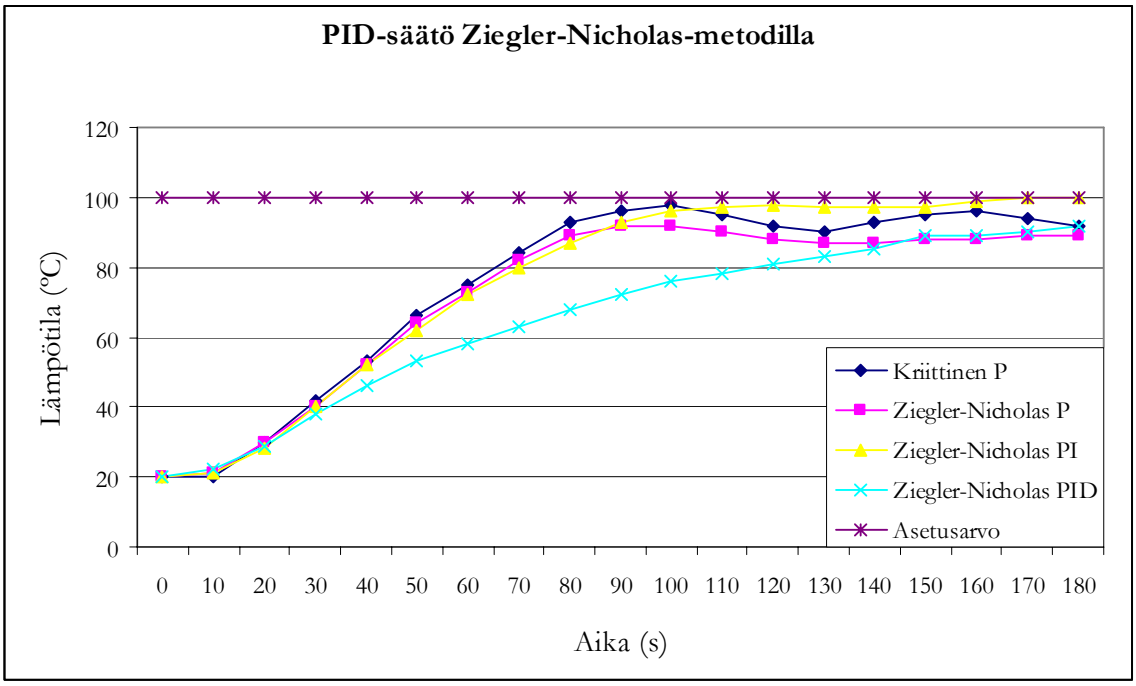
Taulukossa 5 on esitetty saadut tulokset mittauksista.

Taulukko 5. Ziegler-Nicholas-menetelmää käyttäen saadut tulokset.

| Aika (s) | Kriittinen P Lämpötila (°C) | Ziegler-Nicholas P Lämpötila (°C) | Ziegler-Nicholas PI Lämpötila (°C) | Ziegler-Nicholas PID Lämpötila (°C) |
|---|---|---|---|---|
| **0** | 20 | 20 | 20 | 20 |
| **10** | 20 | 21 | 21 | 22 |
| **20** | 30 | 30 | 28 | 29 |
| **30** | 42 | 40 | 40 | 38 |
| **40** | 53 | 52 | 52 | 46 |
| **50** | 66 | 64 | 62 | 53 |
| **60** | 75 | 73 | 72 | 58 |
| **70** | 84 | 82 | 80 | 63 |
| **80** | 93 | 89 | 87 | 68 |
| **90** | 96 | 92 | 93 | 72 |
| **100** | 98 | 92 | 96 | 76 |
| **110** | 95 | 90 | 97 | 78 |
| **120** | 92 | 88 | 98 | 81 |
| **130** | 90 | 87 | 97 | 83 |
| **140** | 93 | 87 | 97 | 85 |
| **150** | 95 | 88 | 97 | 89 |
| **160** | 96 | 88 | 99 | 89 |
| **170** | 94 | 89 | 100 | 90 |
| **180** | 92 | 89 | 100 | 92 |

## 5.3 Tuloksien tarkastelu

Kuvassa 23 on esitetty taulukon 5 tulokset diagrammimuodossa.




Kuva 23. Diagrammikuva säädöistä Ziegler-Nicholas-menetelmällä

Kuvan 23 käyrästä ”Kriittinen P” voidaan nähdä, että prosessimuuttuja alkoi värähdellä jaksonajalla 60 sekuntia. Tällöin P-parametrin arvo oli 100.

Ziegler-Nicholas-menetelmän P-säädössä annettiin P-parametrille arvo 50. Tässä mittauksessa steady-state-virhe jää suhteellisen suureksi. Steady-state-virheen arvo on noin 12 ºC eli mittausalueella noin 15 %. Kuvan käyrä ”Ziegler-Nicholas P” esittää tämän mittauksen.

Ziegler-Nicholas-menetelmän PI-säädössä P-parametrin arvo oli 45 ja I-parametrin arvo 50 s. Tällöin saatiin steady-state-virhe ajettua lähes nollaan. Myös nousuaika oli samaa luokkaa kuin P-säädössä. Tämä on esitetty kuvan käyrässä ”Ziegler-Nicholas PI”.

Lopuksi käytettiin Ziegler-Nicholas-menetelmän PID-säätöä. Tällöin P:n arvo oli 60, I:n arvo 30 s ja D:n arvo 7,5 s. Tässä mittauksessa järjestelmä tuli liian herkäksi häiriöille ja PID-säädön lähtö alkoi värähdellä maksimi- ja minimiarvon välillä. Nousuaika oli huomattavasti suurempi kuin muissa mittauksissa, eikä prosessimuuttuja saavuttanut maksimiarvoaan koko

mittausaikavälillä. D-termin käyttö aiheutti huomattavaa epävakautta järjestelmässä. Tämä on kuvattu kuvan käyrässä ”Ziegler-Nicholas PID”.

Kaikissa mittauksissa kuolleeksi ajaksi saatiin noin 10 sekuntia. Yliampumista ei esiinny missään mittauksessa.

Käyristä voidaan tehdä johtopäätös, että tässä mittausjärjestelmässä PI-säätö oli kiistatta paras. Sillä päästiin suhteellisen nopeasti lähelle asetusarvoa ja prosessimuuttuja myös pysyi siinä. P-säätö toimi vielä auttavasti, mutta steady-state-virhe jäi suureksi. PID-säätö ei sopinut ollenkaan tähän mittausjärjestelmään. Prosessimuuttujassa oli pientä häiriötä, joka saattoi johtua Pt100-lämpötila-anturin kiinnityksestä ja mittausjärjestelmään vaikuttavista häiriölähteistä (virtalähteet). Häiriöiden takia PID-säädön D-parametri aiheutti PID-säädön lähtöön voimakasta vaihtelua. D-parametrin arvon pienentäminen edelleen paransi hieman tilannetta, mutta tälläkään menetelmällä ei päästy lähelle PI-menetelmän suorituskykyä.

# 6 YHTEENVETO

National Instrumentsin CompactRIO on uusi Kajaanin ammattikorkeakoululle hankittu mittaustekniikan laitteisto. Sillä voidaan suorittaa vaativaa prosessinlukua ja -ohjausta. CompactRIO koostuu reaaliaikakontrollerista, jossa on Pentium-luokan prosessori, ja rungosta, jossa on uudelleenohjelmoitava FPGA-piiri sekä runkoon kiinnitettävistä I/O-moduuleista. Tämän insinöörityön tavoitteena oli CompactRIO:n käyttöönotto ja lämpötilanohjaussovelluksen rakentaminen ja testaaminen PID-säätöä käyttäen.

Työtä varten tarvittiin kannettava tietokone, CompactRIO-laite, kaksi I/O-moduulia, kaksi sovitinyksikköä, tehovastus ja Pt100-lämpötila-anturi. Lämpöprosessia simuloitiin tehovastuksella ja Pt100-lämpötila-anturia käytettiin lämpötilan lukemisessa. Sovitinyksiköt kytkettiin I/O-moduuleihin, jotka asetettiin CompactRIO:n runkoon. Pt100-lämpötila-anturi oli kytketty tulosovittimeen ja tehovastus oli kytketty lähtösovittimeen. Näin saatiin rakennettua suljettu säätöjärjestelmä.

CompactRIO:lle tehtiin kaksi ohjelmaa LabVIEW:illä. FPGA-piirille tehtiin yksi ohjelma tiedon lukua ja kirjoitusta varten, ja reaaliaikakontrollerille tehtiin yksi ohjelma tiedon prosessointia varten. Reaaliaikakontrollerin ohjelmassa tieto muutettiin ensin desimaaliluvuksi (jännitteeksi) ja sitten lämpötilaksi. Tämän jälkeen suoritettiin PID-säätö annettujen parametrien avulla. Lisäksi tietokoneelle tehtiin isäntäohjelma, jolla voidaan seurata ja kontrolloida prosessin kulkua.

Järjestelmän testaus suoritettiin tekemällä neljä testimittausta. Ensin tehtiin mittaus, jotta löydettiin kriittinen vahvistus ja tämän jälkeen tehtiin kolme mittausta Ziegler-Nicholas-menetelmän avulla. Mittauksista voitiin todeta, että Ziegler-Nicholas-menetelmän PI-säätö toimi parhaiten tässä järjestelmässä.

Työn suorituksessa oli myös ongelmia. Suurin ongelma työssä oli se, että käytettävä vastus ei ollut riittävän nopea lämpenemään ja jäähtymään. Se jouduttiin vaihtamaan aivan työn loppuvaiheessa massaltaan ja vastusarvoltaan pienempään vastukseen. Myös I/O-moduulien kanssa oli ongelmia. Digitaalinen I/O-moduuli särkyi kesken työn, ja tämän takia ohjaukseen jouduttiin käyttämään analogista I/O-moduulia. Häiriöitä oli myös runsaasti, ja ne vaikuttivat mittaustuloksiin. Häiriöt ilmeisesti johtuivat erilaisista virtalähteistä, joita oli mittauspaikan läheisyydessä.

Ongelmista huolimatta mittausjärjestelmä saatiin lopulta toimimaan odotetulla tavalla ja halutut testimittaukset saatiin suoritettua. Testimittaukset olivat kuitenkin vain säätöjärjestelmän lopullinen toimivuustesti. Työn päätavoite oli toimivan mittausjärjestelmän rakentaminen ja tiedonvälityksen toteuttaminen.

Jatkossa lämpötilanohjausprosessin ohjelmistoa ja laitteistoa voisi kehittää siihen suuntaan, että lämpötilan muutoksista tulisi nopeampia, varsinkin alaspäin. Lähtöön voisi kiinnittää esimerkiksi tuulettimen, joka käynnistyy, kun asetusarvo ylitetään. Myös vastuksen massaa voisi edelleen pienentää lämpötilan muutoksen nopeuttamiseksi.

Kokonaisuudessa työ osoitti CompactRIO:n toimivuuden mittausjärjestelmissä ja prosessinohjauksessa. CompactRIO:lla on lukuisia käyttömahdollisuuksia erilaisissa mittausjärjestelmissä.

# LÄHTEET


1 National Instruments. Real-Time Tutorial [WWW-dokumentti]
http://zone.ni.com/devzone/cda/tut/p/id/3938

2 Forsten, J., Karius, J. & Korpela, E. Laskentatehoa verkotetulla klusterilla. Prosessori –lehti 1/2005 s. 31.

3 National Instruments. FPGA-Based Control: Millions of Transistors at Your Command (FAQ) [WWW-dokumentti]
http://zone.ni.com/devzone/cda/tut/p/id/3357

4 National Instruments. PID Theory Explained [WWW-dokumentti]
http://zone.ni.com/devzone/cda/tut/p/id/3782

5 Wikipedia. PID-controller. [WWW-dokumentti]

http://en.wikipedia.org/wiki/PID_control

6 National Instruments. NI CompactRIO Control and aquisition System [WWW-dokumentti]
http://www.ni.com/compactrio/whatis.htm

7 National Instruments. NI cRIO-9004. [WWW-dokumentti]
http://sine.ni.com/nips/cds/view/p/lang/en/nid/14161

8 National Instruments. NI cRIO-9104. [WWW-dokumentti]
http://sine.ni.com/nips/cds/view/p/lang/en/nid/14159

9 National Instruments. CompactRIO I/O Modules. [WWW-dokumentti]
http://sine.ni.com/nips/cds/view/p/lang/en/nid/14147

10 National Instruments. NI9215. [WWW-dokumentti]
http://sine.ni.com/nips/cds/view/p/lang/en/nid/14166

11 National Instruments. NI9263. [WWW-dokumentti]
http://sine.ni.com/nips/cds/view/p/lang/en/nid/202557

12 Wikipedia. LabVIEW. [WWW-dokumentti]

http://en.wikipedia.org/wiki/Labview

13 National Instruments. LabVIEW FPGA Module User Manual [WWW-dokumentti]
http://www.ni.com/pdf/manuals/370690b.pdf

VUOKAAVIO

Vuokaaviossa esitetään yksittäisen luku- ja ohjauskierroksen suorittaminen järjestelmässä.